MSX POCKET BANK 4

ポケットバンク

鳴らして、聞いて、おぼえる音作り

マイコン・サウンドパック

工藤賢司



ズシン、ビシッ、バシッ
どんな音でも思いのまま
すぐに使えるサウンドデータ集

アスキー出版局

MSX POCKET BANK 4

マイコン・サウンドパック

工藤賢司

アスキー出版局

INTRODUCTION

「カキィーン」「ズコン」「パシッ」「ザザッ」「ガクッ」「ポイ」「ゴボゴボ」「ビュオオオオ」「ビリビリ」「カポン」「ウィーン」「ボムン」「ドカドカ」「ギロッ」「ザバーッ」…………

読者のみんなは一体これ，何の音だと思うかな？

実は，全部某少年マンガ誌の中から拾った効果音なんだ．マンガ家の人たちは絵の中で音を表現しようとして，いろんなカタカナの擬音を苦心して作り出してきた．動きと迫力を伝えるためには，どうしてもカッコイイ擬音を使うことが欠かせないのだ．

ところで，パソコンの世界でもテレビゲームを中心に，音楽や効果音は不可欠の要素となっている．MSXをはじめとして多くのパソコンには，超小型のサウンド・ジェネレータが内蔵されていて，ゲームセンターと同じ音を自由に作り出せる．

この本は，MSXパソコンを利用して，様々な効果音を作ってみようというものだ．BASICの命令を通じての「音作り」のノウハウや，たくさんのサウンドデータ・サンプルを満載してある．また，効果音を作成・編集するための便利な「サウンド・エディタ・プログラム」や，実際のゲームの中での音の使い方なども紹介する．

マイコンをシンセサイザー代わりに使うための，「マイコン・サウンド・パッケージ」，一冊丸ごとじっくりと耳で楽しんで欲しい．

なお，本書の４章に掲載した「サウンド・エディタ２」については，アスキー書籍編集部の金光君にご協力いただいた．この場を借りて厚くお礼申し上げる．

昭和59年３月　　工藤賢司

表紙デザイン　HIT STUDIO
表紙イラストレーション　日比野克彦
本文イラストレーション　石渡希和子

CONTENTS

BLOCK ONE 1 パソコンと効果音

その昔，パソコンには音がなかった．

いまでこそ，パソコン・ショップに足を運ぶと，たくさんのパソコンがデモ演奏を鳴りひびかせているけれども，はじめのころは「BEEP(ビープ)」という命令による「ピーッ」という音しか出せなかったものだ．

けれども，パソコンが数多く世に出まわって，ゲームなどに利用されることが多くなると，しだいに歌のひとつも歌えるような機種が増えてきた．

やがてパソコンは，音楽や効果音の世界で，大活躍をはじめるようになったのである．

1-1 PSGは小さなシンセサイザー

多くのパソコンが音を出せるようになったのは，通称「PSG(プログラマブル・サウンド・ジェネレータ)」と呼ばれる音出し専用のLSIを内蔵しているからだ．このPSGというやつ，ガラは小さいけれども，立派なシンセサイザーの一員と考えていいだろう．なにしろこれ1個で，ずいぶんいろんな音のバリエーションが楽しめる．

MSXにも，PSGがひとつ取りつけられていて，おかげで3重和音と1つのノイズ(雑音)を出すことができる．

もともとPSGは，やはりビデオゲームのために開発されたLSIだった．現在でもほとんどの街のゲームセンターでは，このPSGが音出しの役目を引き受けている．

思い起こせば，「スペース・インベーダー」や「風船わり」(知ってるかな？)と

いった古典的なゲームの時代から，効果音は切っても切り離せない存在だった．そして，パソコン利用の一分野として，ビデオゲームが大きくクローズ・アップされてくるとともに，この音出し機能もそのまま受けつがれる形となったのだ．

音のないゲームがいかに味気ないものかは，ちょっとテレビのボリュームを消して，MSX でゲームをやってみればすぐにわかるだろう．

実際，ボクたち人間は，「グツグツ」とナベの煮える音を聞くだけで，ゴクリとツバを飲みこみ，「カサカサ」と枯葉の音を聞くだけで，もの悲しい気分におちいったりする．「百聞は一見にしかず」などと言うけれども，ボクたちはふだん，目でみる以上に耳からも，ずいぶんといろいろなことを感じとっているのである．

ビデオゲームなども半分は耳で楽しんでいる，と言ってよいだろう．

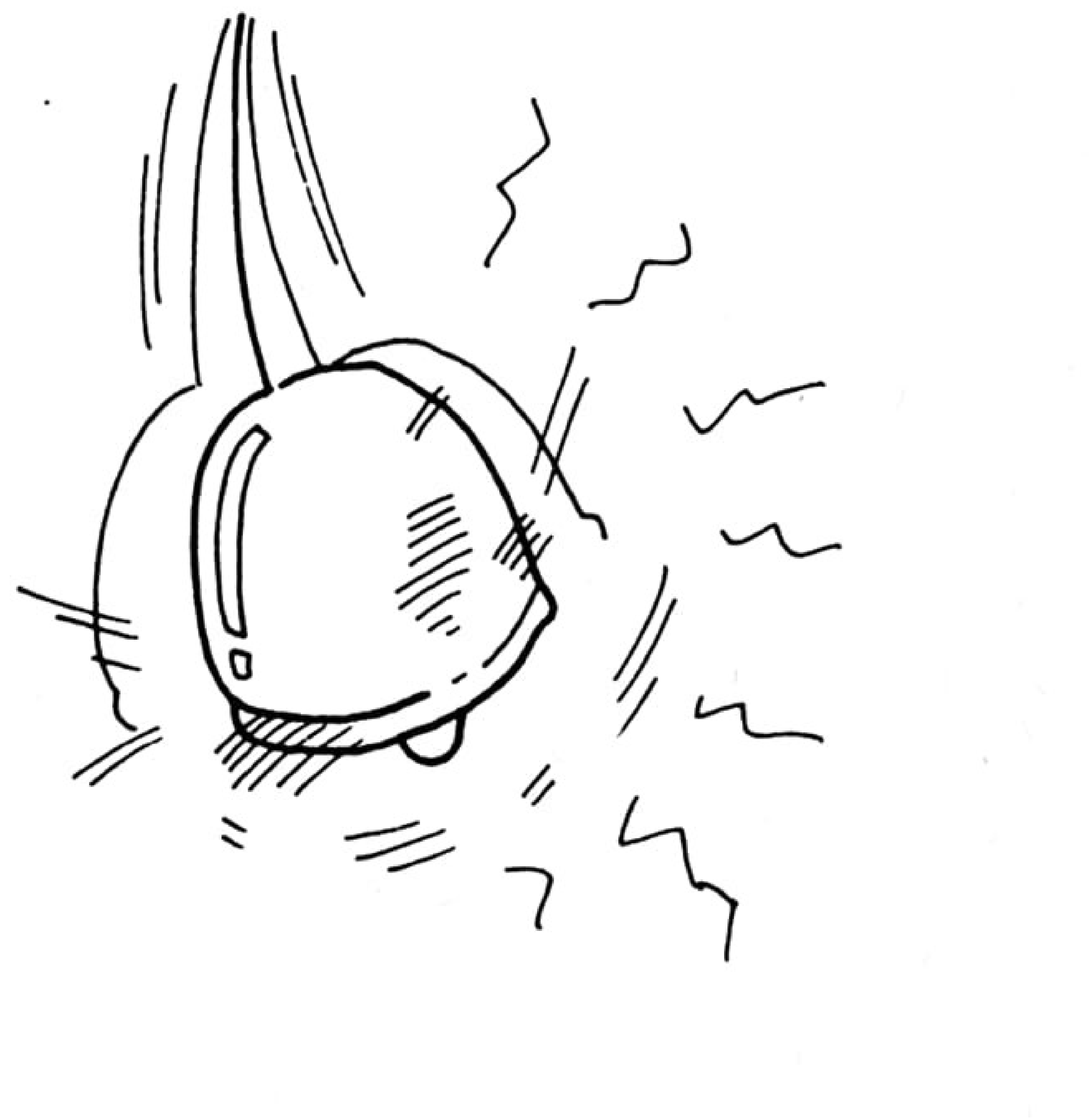
プレイかサウンドか？ 1-2

さて, MSX の BASIC には, 音を出すための命令として,「PLAY（プレイ）」と「SOUND（サウンド）」の 2 つが用意されている．この 2 つは，どちらも PSG をコントロールして音を出す点では同じなのだが，少々性格がちがっている．

「PLAY」のほうは，いわば音楽専用の命令で，オタマジャクシで書かれた楽譜をパソコンで演奏するためのものだ．

ためしにちょっと実行してみよう．

```
10 PLAY "o4cdefgabo5c"
```



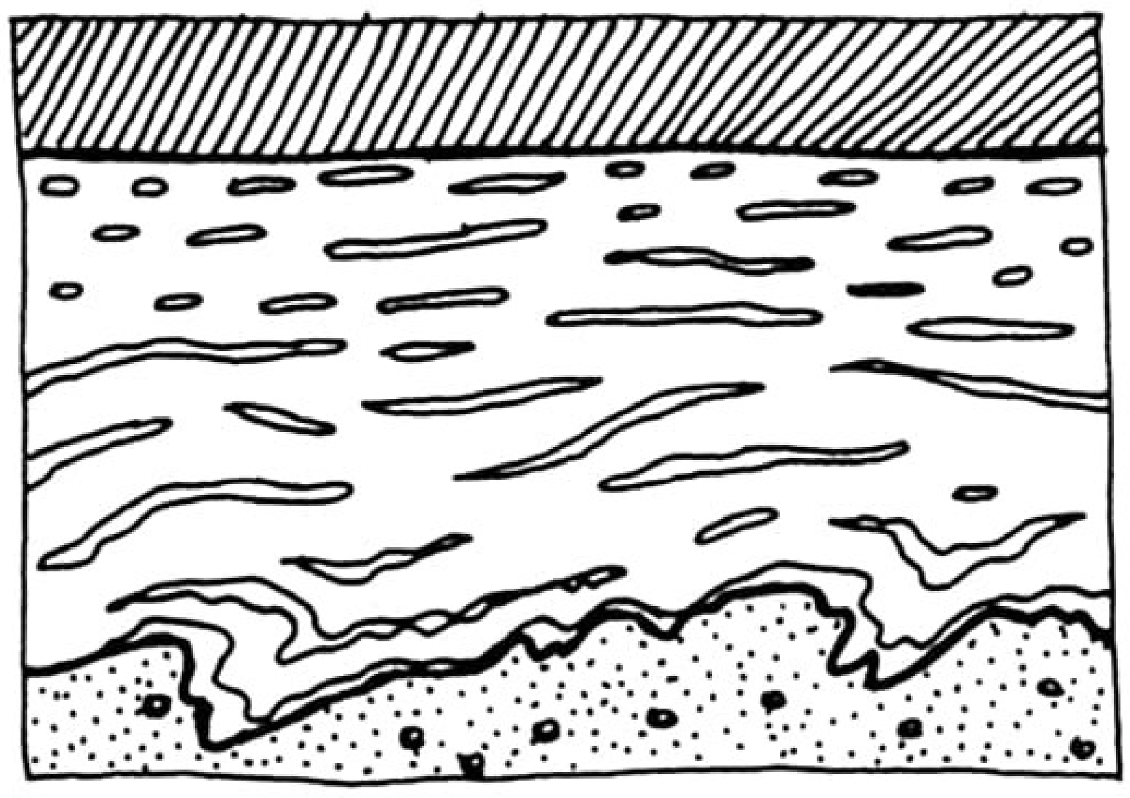


PLAY 命令では，このように，「ミュージック・マクロ」と呼ばれるパソコン用の楽譜(これでも楽譜なのだ)を使って音楽を演奏する．けれども，この命令では，「ド(C)」と「ド#(C#)」の中間の音を出したり，「ザー」というノイズを出したりすることはできない．

これに対して，「SOUND」のほうは，直接 PSG の機能と深く関係しており，ミニ・シンセサイザーとしての PSG の実力を十二分に引き出すことができる．

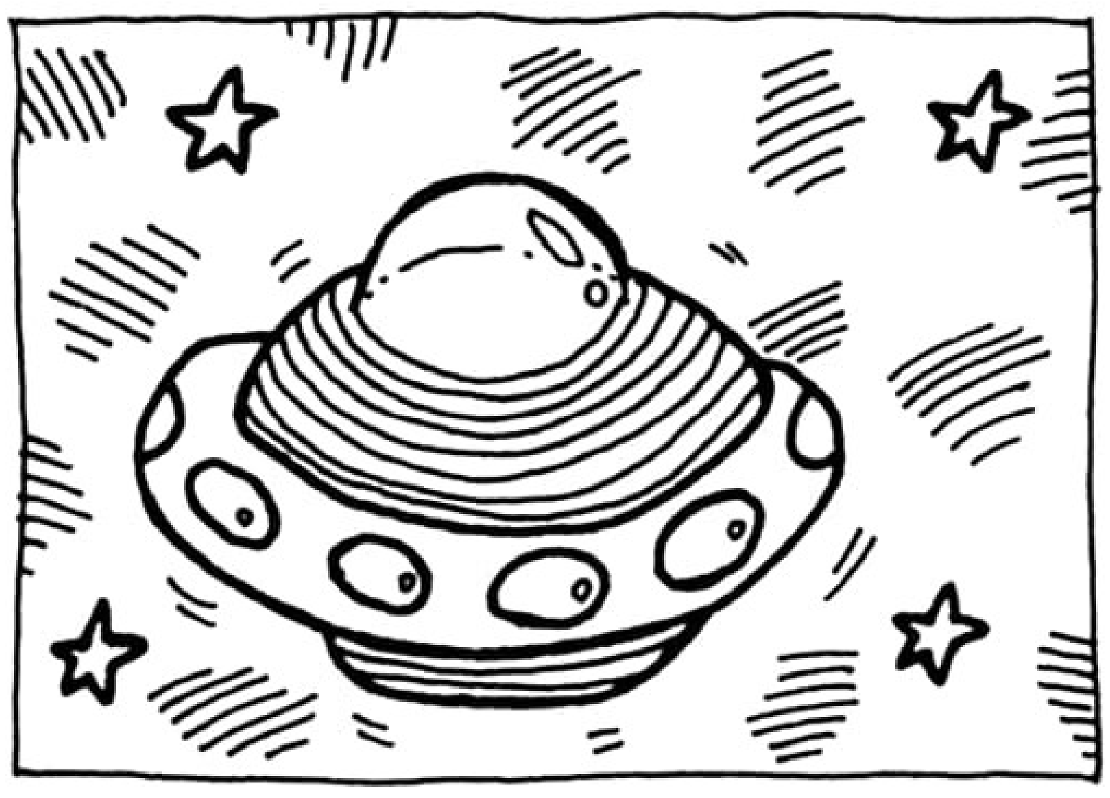
```
10 SOUND 7,55
20 SOUND 6,15
30 SOUND 8,16
40 SOUND 11,103
50 SOUND 12,91
60 SOUND 13,14
```

① 「波」の音



```
10 SOUND 7,60
20 SOUND 8,15:SOUND 9,15
30 SOUND 1,0:SOUND 3,0
40 FOR I=0 TO 1 STEP .1
50  SOUND 0,40*I^2+100
60  SOUND 2,40*I^2+70
70 NEXT I
80 GOTO 40
```

② 「UFO」の音



上のリストのうち，①は「波の音」，②は「UFO の飛行音」だ．

このように，SOUND 命令を使えば，爆発音だの，エンジン音だのといった，ビデオゲームに欠かせない効果音を自在に出すことができる．

この本では 2 つの音出し命令のうち，「SOUND」のほうに的をしぼって，くわしく説明する．SOUND 命令の使い方を覚えるには，いろいろなパターンの音を出してみるのがいちばんなので，できるだけたくさんのサンプル・データを載せておいた．本書を読み終わるころには，きっと PSG を自由自在にあやつれるようになっているだろう．

なお，PLAY 命令のほうについてくわしく知りたい人は，MSX ポケットバンク・シリーズの「マイコン・ジュークボックス」のほうを読んでいただきたい．

BLOCK TWO 2 「音」ってなんだ?

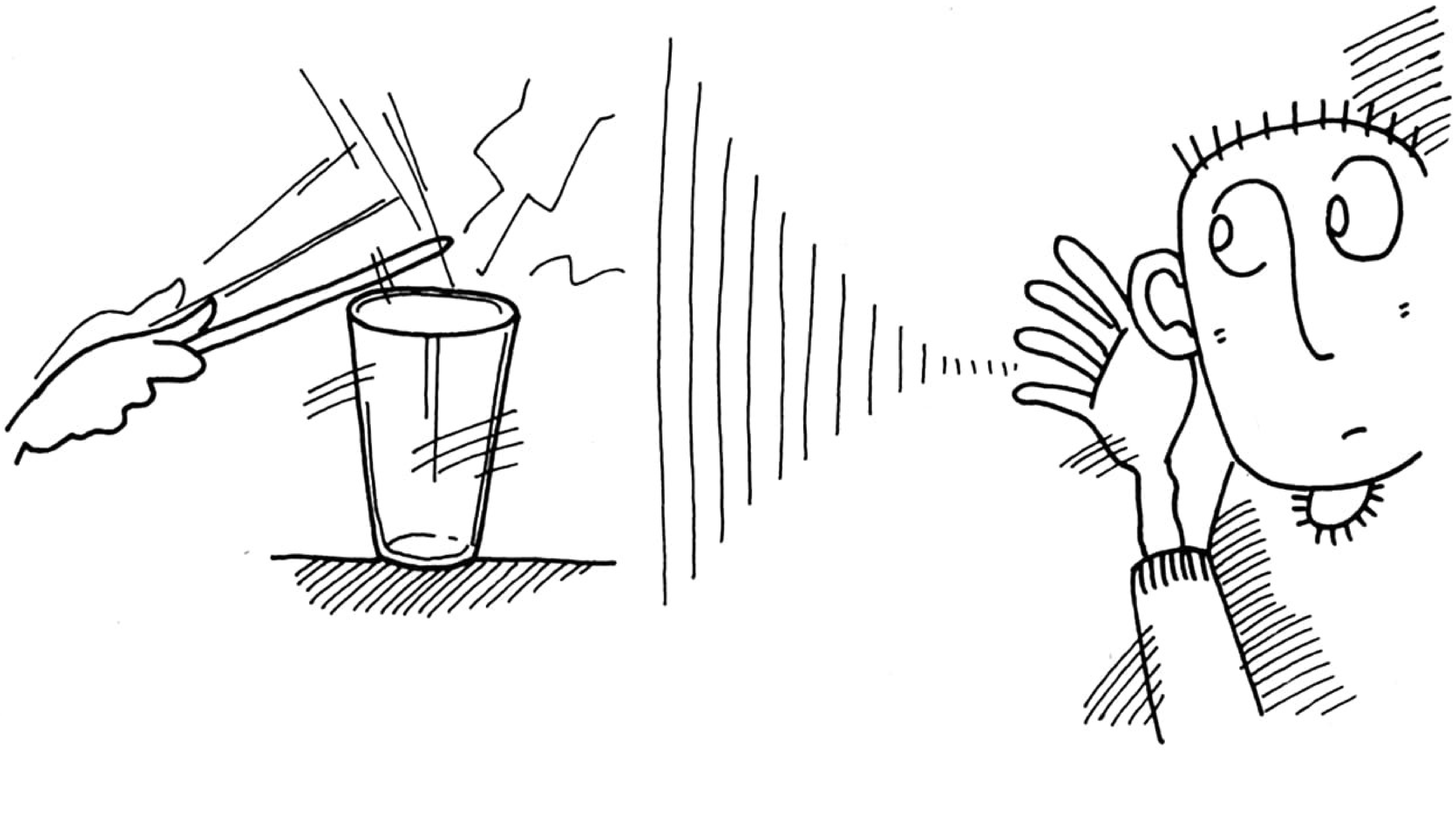
さて，これまで気楽に「音を作る」なんて言ってきたが，作るというからには「音」の正体や性質について，少し知っておく必要がある．

「音とは空気の振動である」なーんて，たしか小学校の理科の時間に習った気もするが，いったいぜんたいピアノの音とオルガンの音は，どうちがうのか，などということについては，案外忘れてしまった人も多いんじゃないかな．

そこでちょっと，音について，おさらいをしておくことにしよう．

音が耳に伝わるまで 2-1

いまかりに，ガラスのコップを棒でたたいたときのことを考えてみる．「ピンピン」と軽い感じの澄んだ音がするだろう．この音が人間の耳に伝わるまでの経路をたどってみると，だいたいこんな具合だ．



コップをたたくと，まずコップの側面がふるえる．このふるえはコップのまわりの空気を振動させて，空気の波を巻き起こす．

波は，どんどんと空気中を伝わっていき，その一部は耳の中の鼓膜にまで達する．すると，鼓膜のふるえを聴覚神経が感じとって，脳ミソに「音が聞こえたゾゥ」と知らせてくれるわけだ．

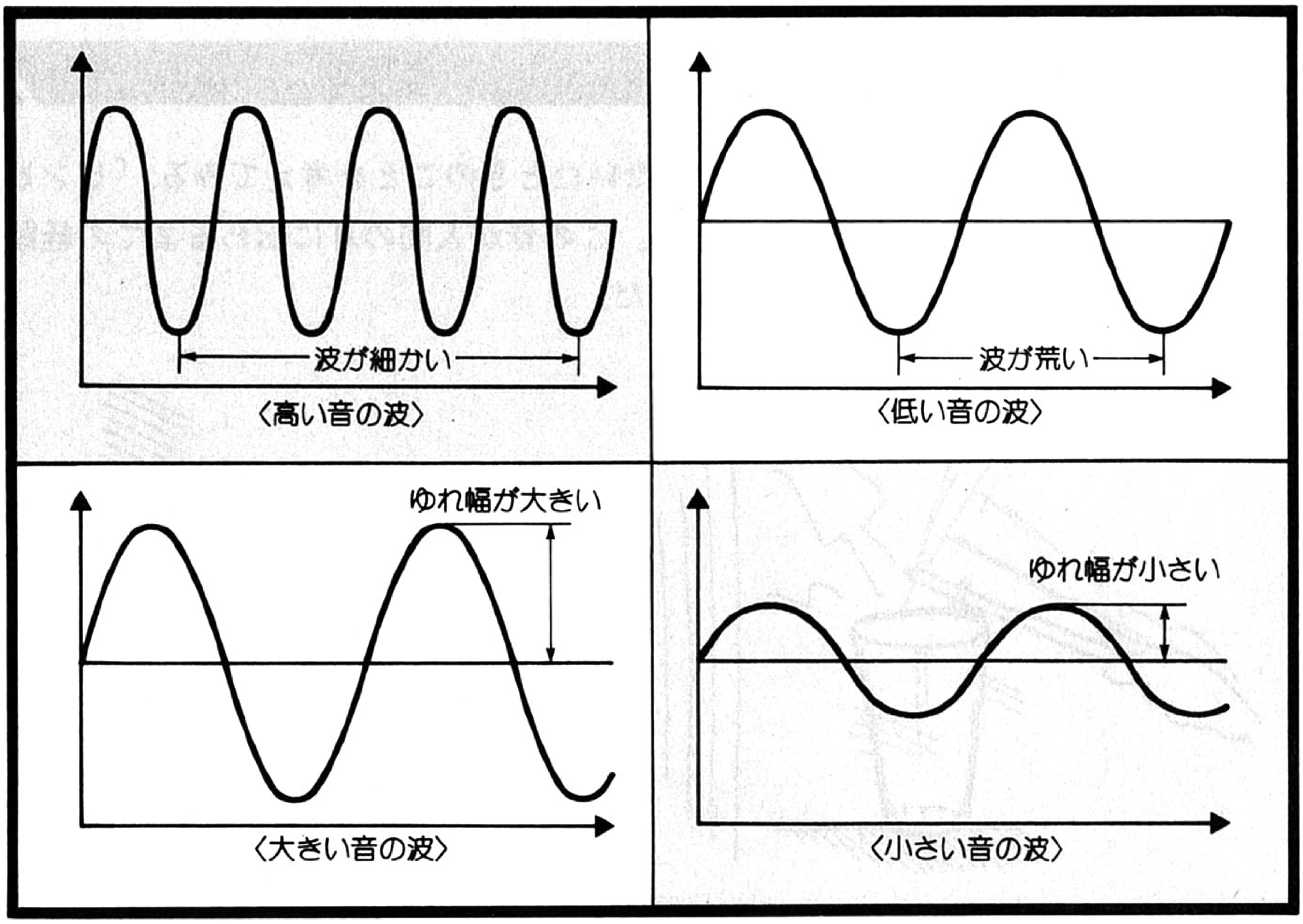
このように，音というのは，空気の中を伝わる波の一種なのである．宇宙空間で音が聞こえないのは，かんじんの音を伝える空気が存在しないからだ．

2-2 音の高さと強さ

ところで，ひとくちにコップの音といっても，小さいコップをたたいたときには高い音がするし，大きなコップのときには低い音がする．また，強くたたけば大きな音が，弱くたたけば小さな音がする．

こうした音のちがいは，「空気の波」を目で見てわかるように，横軸に時間をとってグラフ化してみるとよい．

図 2 - 1



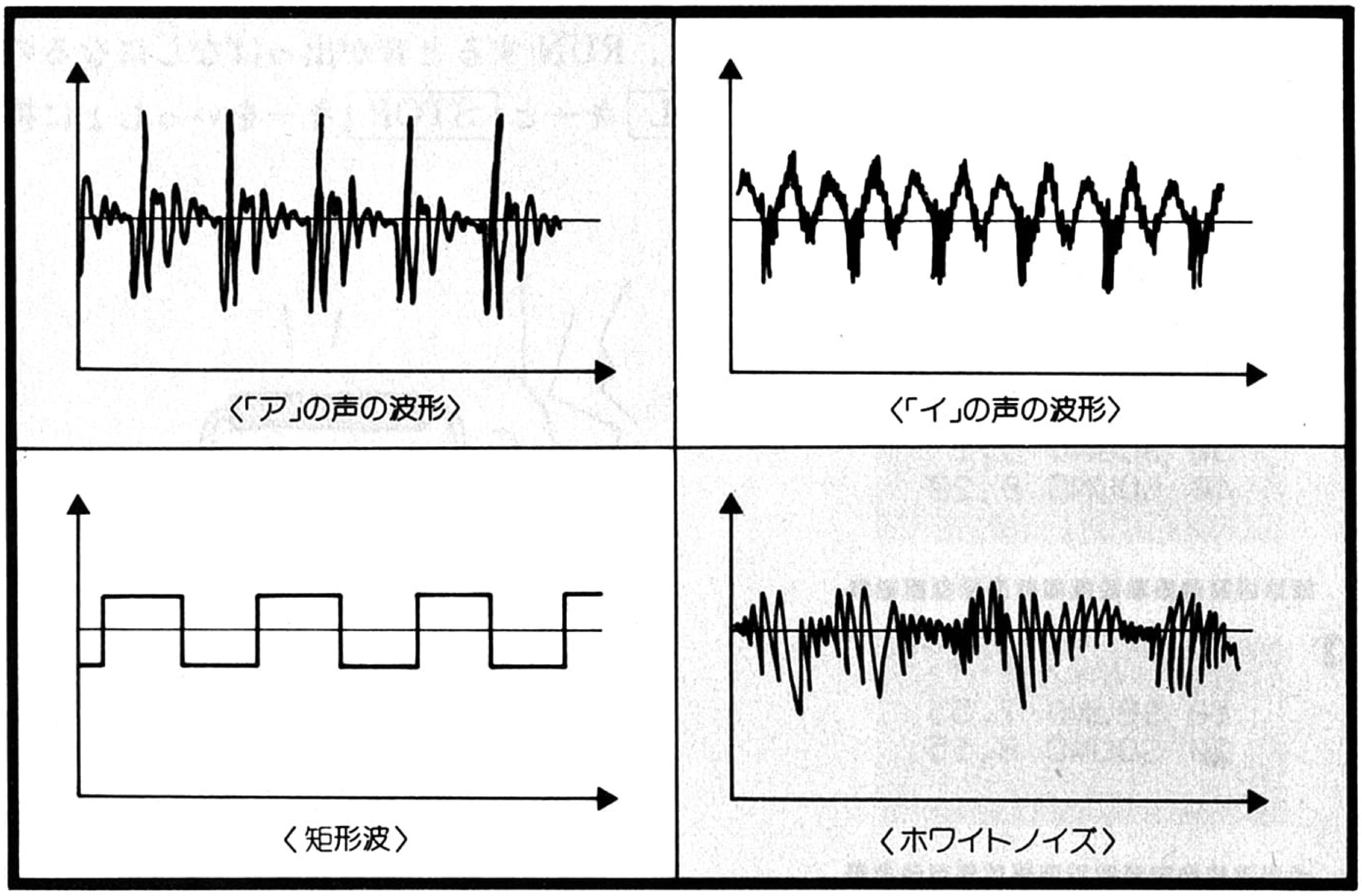
たとえば，**図2－1**のように，「高い音」は波が細かく，「周波数」(1秒間に何回波が振動するか)が高いのに対して，「低い音」はゆったりとしていて，「周波数」が低い．つまり，音の高さをきちんと表わすには，周波数(単位はヘルツ，Hz)で指定してやればよい．人間の耳に聞こえる音は，個人差もあるが，ふつう20ヘルツから20,000ヘルツくらいまでだと言われている．

いっぽう，音の「強さ」のほうは，波の「振幅」として表わせる．「振幅」というのは，波の山から谷までのゆれ幅のことだ．大きい音ほどこの幅が広いことになる．ちなみに音の強さを表わすには，デジベル(dB)という言葉を使う．もしかしたら，オーディオなんかのカタログで，お目にかかったことがあるかもしれない．

波形と音色 2-3

さて，音の性質を表わす要素としては，「高さ」，「強さ」のほかに「音色」というのがある．じつは，音の波というのは，左ページの図のように，単純なサイン・カーブの形をしているわけではない．いろいろな周波数の波が重なりあって，むしろ複雑な波形となっている．

図2-2



ボクたちの耳に聞こえる音が，たとえ同じ高さの音でも「これは，ピアノの音だ」とか，「これは，ハーモニカだ」とかの区別がつくのは，それぞれの音に固有な波の形のちがいによるものだ．たとえば，人間の声なんかは，めっぽう複雑なほうの代表選手のようなもので，「ア」と「イ」の音の波は，それぞれちがう形をしている(前ページ図)．

いわゆるシンセサイザーなるメカは，こうした複雑な波形のちがいを自由に作りだすことができる．シンセサイザーが，右は猫の鳴き声から，左はロケットの爆発音まで，すべて１台の機械でこなせるのは，いろいろな波形を電気的に合成する機能をそなえているからなのだ．

ところで，わがMSXに組みこまれているPSGはというと，そこは超ミニ・シンセの悲しさか，出せる波形は２つしかない．

ひとつは，矩形波(くけい)(山や谷が四角い形をしている)の音で，これは人間の耳には「ピー」とか「ポー」というふうに聞こえる．もうひとつは，「ザー」という音で，ホワイトノイズと呼ばれている．テレビなんかの空きチャンネルから聞こえてくる雑音と同じものだ．

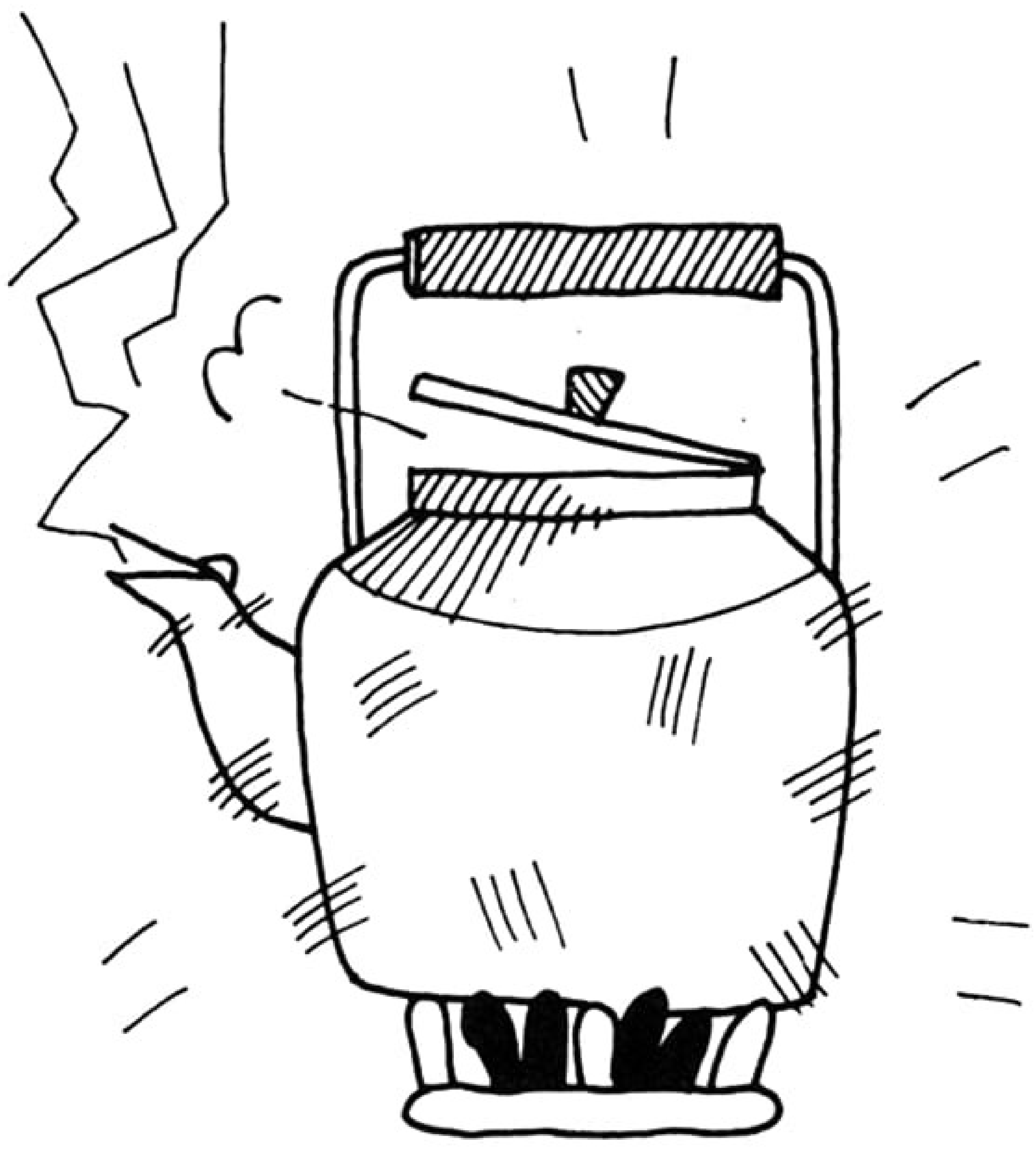
ここでちょっと，２つの音を聞きくらべてみることにしよう．次にあげるリストの①は「ピー」という音を，リストの②は「ザー」という音をパソコンから出すプログラムだ．(２つのプログラムとも，RUNすると音が出っぱなしになるので，止めるときには，例によって CTRL キーと STOP キーをいっしょに押す．)

①

```
10 SOUND 7,62
20 SOUND 8,15
30 SOUND 1,1
40 SOUND 0,28
```

②

```
10 SOUND 7,55
20 SOUND 8,15
```



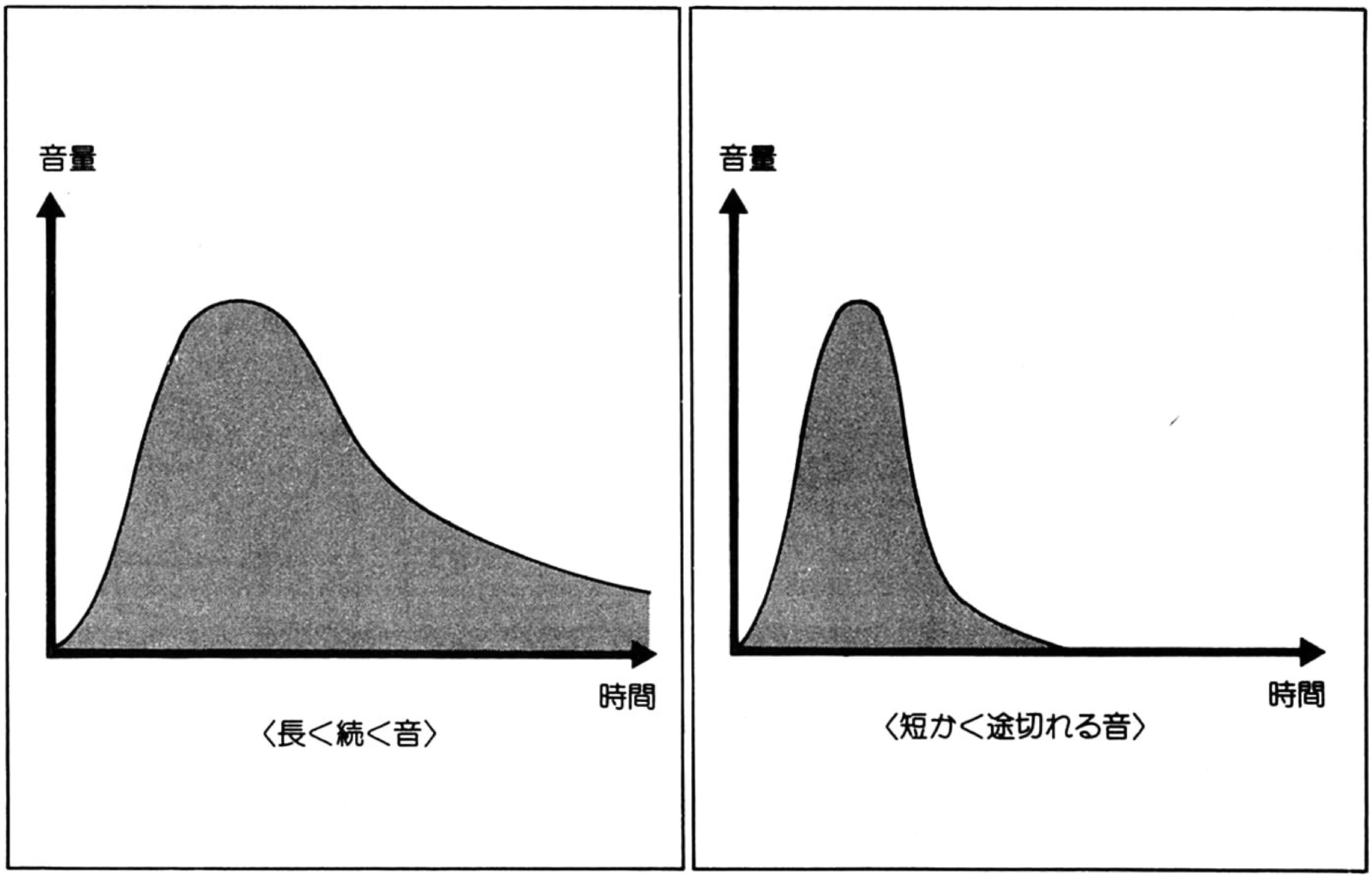
けれども，音の種類が２つしかないからといって，がっかりする必要はない．ボクらの耳に，ある音がそれ「らしく」聞こえるためには，これまでにあげた「高さ」「強さ」「音色」の３つのほかに，もうひとつ，「エンベロープ」という大事な要素があるのだ．

長い音・短い音 2-4

「エンベロープ」とは，耳なれない言葉だが，要するに音の立ち上がり方や，消え去り方のパターンを表わすものだ．

たとえば，お寺の鐘の音は，「ゴーン・ウォン・ウォン」と長く余韻がのこるが，ものさしで机をひっぱたいたときの音などは，「ピシリ！」とすぐに途切れてしまう．このちがいを，音が鳴りはじめてからの音量の変化としてグラフ化してみると，**図２－３**のようになる．

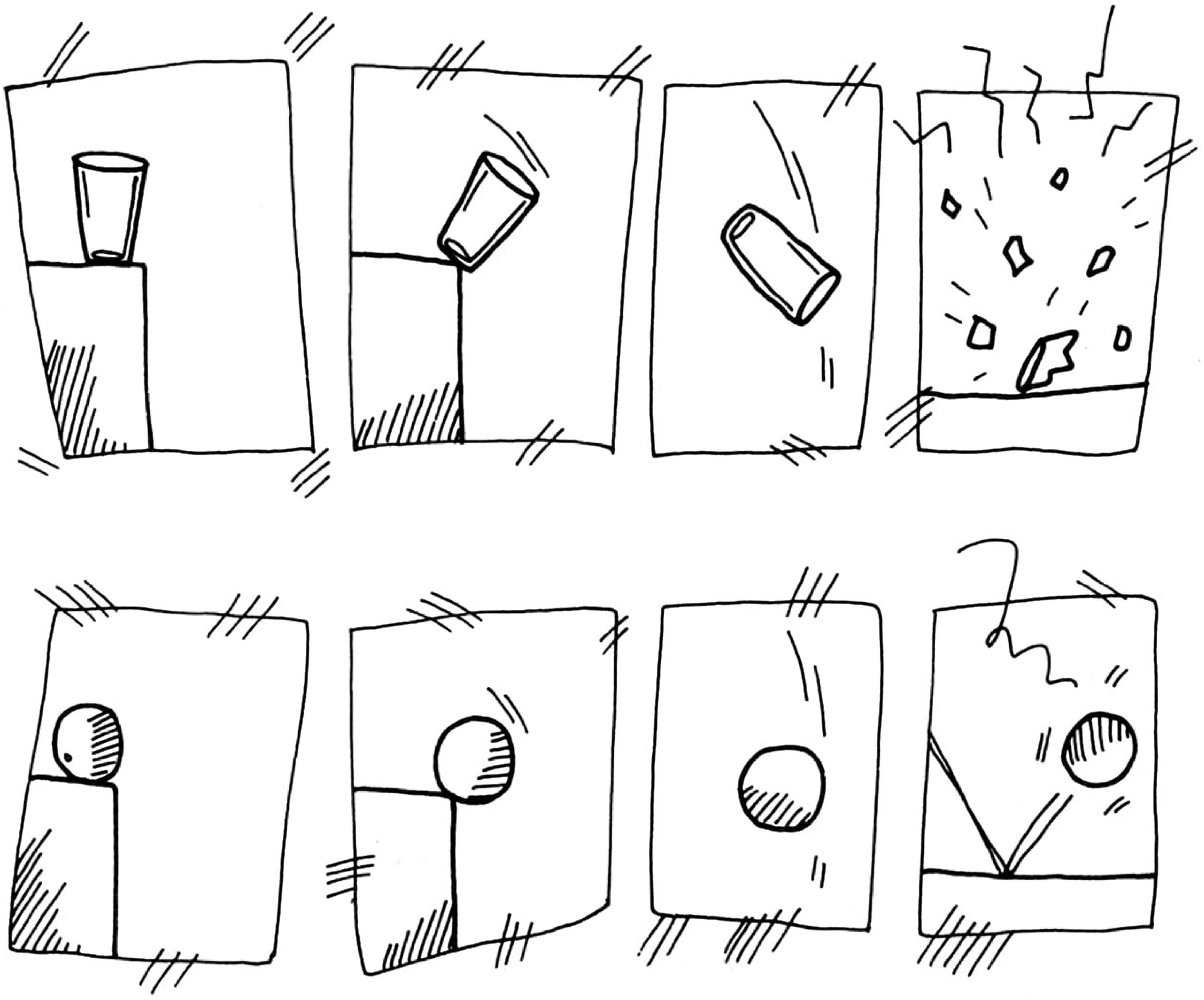
図２-３



「エンベロープ」は，音の雰囲気を出すためには欠かせないもので，これがちがうと，まったく同じ高さ，強さ，音色でも，ずいぶんとちがった感じの音に聞こえる．

パソコン内蔵のPSGの場合には，このエンベロープの型をかなり細かく指定できるしくみになっていて，いろいろな音のバラエティを作り出すことができる．悪くいえばPSGは，出せる音の種類の少なさをエンベロープでごまかしている，いや，健気(けなげ)にもカバーしている，と言うべきだろう．なにしろ手のひらにのってしまう超小型シンセなのだから，ン十万円もするシンセと比較しては，かわいそうというものだ．

くわしい説明はあとにまわすが，とりあえず「エンベロープ」という言葉を頭に入れておこう．



というわけで，音の正体についての基本的な知識はひととおりそろった．ある音をPSGから出すためには，「高さ」「強さ」「音色」「エンベロープ」の4つを，PSGに指示してやればよい．それでは，次の章からPSGとSOUND命令について，くわしく述べることにしよう．

BLOCK THREE 3 PSGとSOUND命令

この章では，PSGのしくみとSOUND命令の使い方について説明する．前のところでも少しふれたように，SOUND命令は，いろいろな指示を音出し専用のLSIであるPSGに対して，直接書きこむ命令だ．そこで，SOUND命令を使って望みの音を出すためには，あらかじめPSGの構造や働きについて，ある程度理解しておかなければならない．

PSGの内部構造 3-1

まず，PSGの内部で音の信号が作り出されてから出力にいたるまでの，おおまかな道筋について見ておこう．

次ページの**図3－1**のように，PSGには3つのトーン（「ピー」という音）発生装置と，1つのノイズ（「ザー」という音）発生装置の，合計4つの音源が組みこまれている．これに対して，出力チャンネルのほうは，A，B，C，の3つがある．

各チャンネルの出力は，最終的にはひとつにまとめられて，テレビやラジカセなどのアンプ類に送り出されるのだが，もちろんその前の段階では，音の高さや強さなどをチャンネルごとに別々に決めることができる．

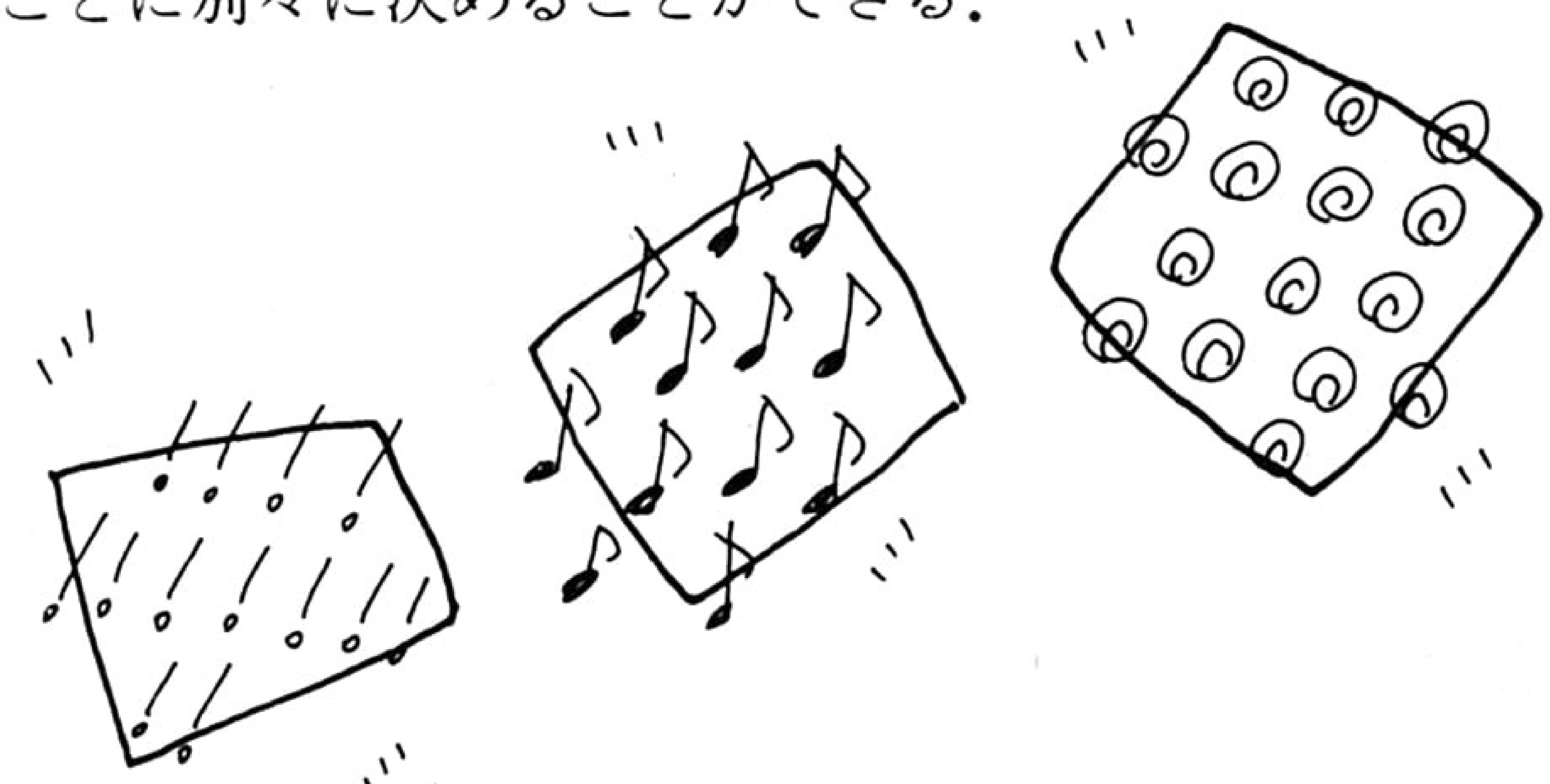
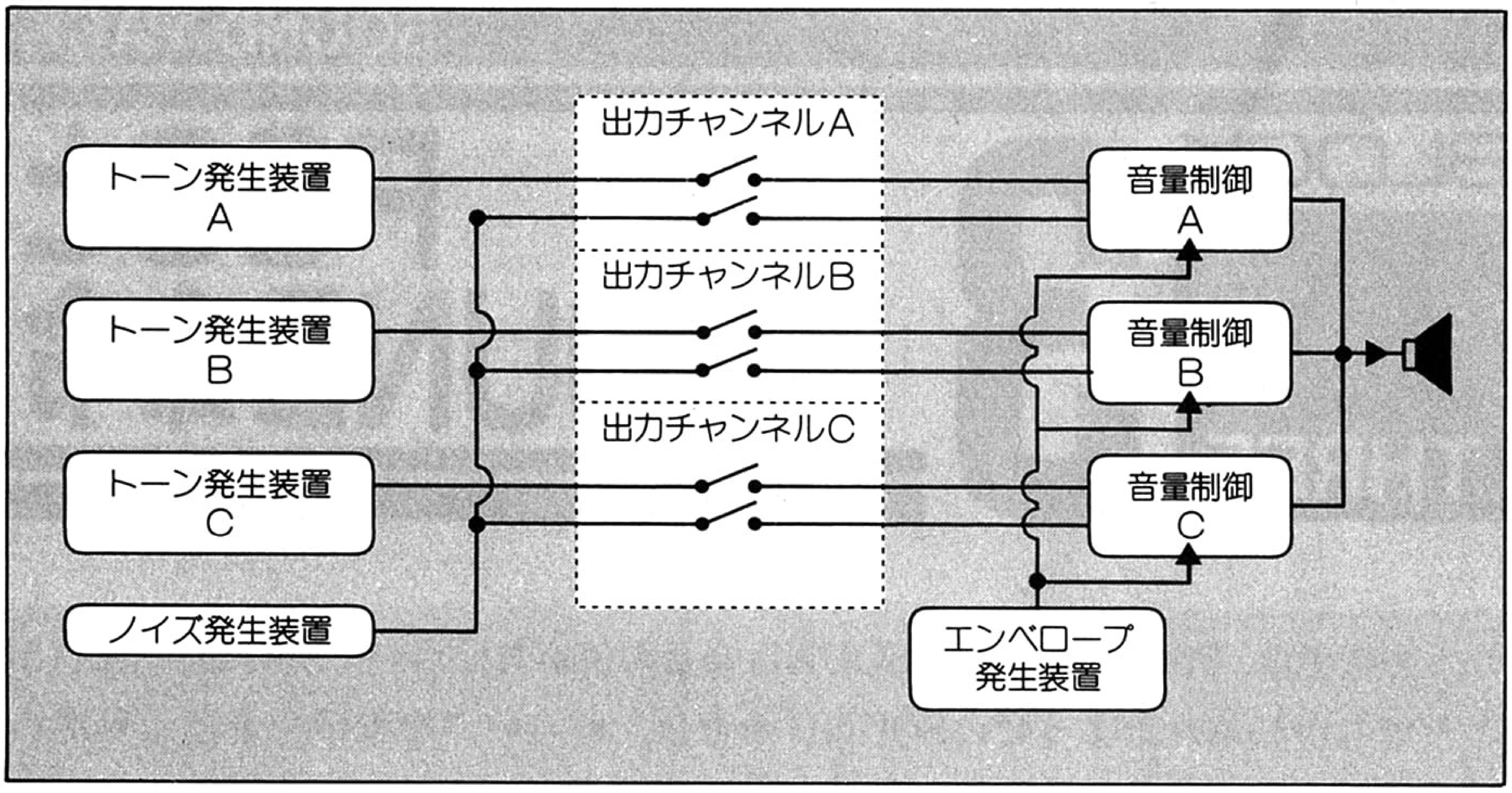


図3－1

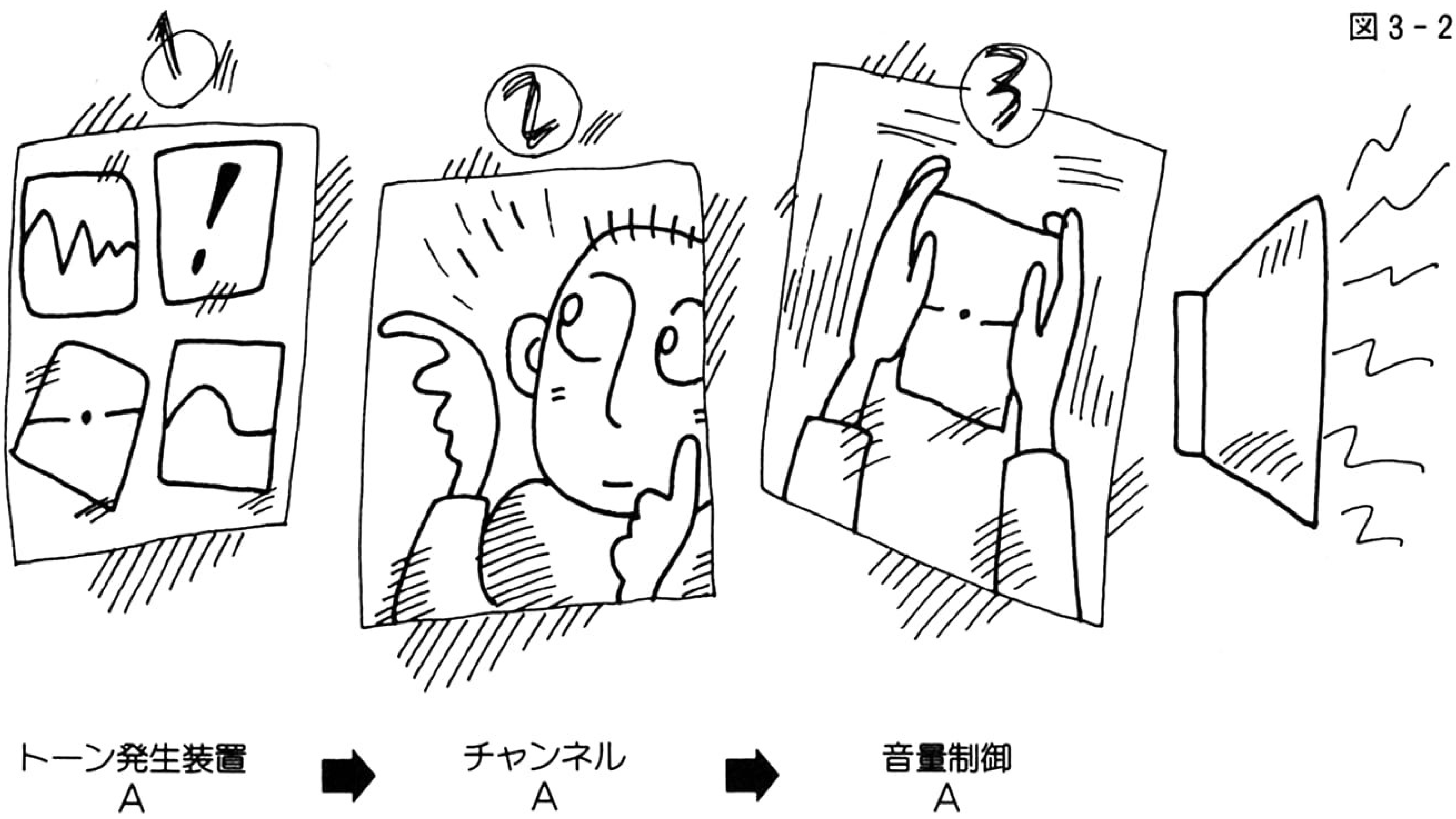


たとえば，チャンネルAからトーンを出すには，**図3－2**のような経路をたどることになる．

もし，和音を出したければ，チャンネルA，B，C，のそれぞれのトーン発振器をONにして，好きな周波数をチャンネルごとにセットしてやればよい．高さのちがう3つの音が入り混じって，和音(ときには不協和音)が聞こえてくる．

また，音源としてトーンの代わりにノイズ発振器を選べば，「ザアザア」と土砂降りの雨のごとく，雑音がスピーカーからあふれ出てくることになる．

図3－2



で，こうしたチャンネルごとの音源のON／OFFや，音の高さ，大きさなどを決めるのが，「SOUND命令」というわけなのだ．おおかたのパソコン命令の例にもれず，これらの指示は，すべて数字であたえてやる必要がある．

テレビならチャンネルを変えるには，チャンネルつまみをまわせばよい．ピアノなら高い音を出すには，高いほうの鍵盤をたたけばよい．しかし，パソコンではなにもかにも，

SOUND 〔いくつ〕，〔いくつ〕

で指定してやらなければならないわけだ．

まあ，理屈さえ飲みこんでしまえば，そうむずかしいものでもない．それに，数字で指定するということは，それなりに便利なことでもあるのだ．それは，おっつけわかってもらえると思う．

14コのレジスタ 3-2

次に，PSGの内部にあるレジスタの説明に移ろう．

レジスタというのは，簡単に言えば，SOUND命令から数字を受け取るための「郵便箱」だと思えばよい．PSGの内部には，全部で14コ(！)のレジスタがあって，**表1**のようにそれぞれちがう役割を受け持っている．そして，各レジスタには，0番から13番までの番号がついている．

表1

レジスタ0	チャンネルAの周波数	レジスタ7	各チャンネルの音源の選択
レジスタ1	チャンネルAの周波数	レジスタ8	チャンネルAの音量
レジスタ2	チャンネルBの周波数	レジスタ9	チャンネルBの音量
レジスタ3	チャンネルBの周波数	レジスタ10	チャンネルCの音量
レジスタ4	チャンネルCの周波数	レジスタ11	エンベロープの周期
レジスタ5	チャンネルCの周波数	レジスタ12	エンベロープの周期
レジスタ6	ノイズの周波数	レジスタ13	エンベロープの種類

PSGは，この郵便箱に送られてきた数字を見て，それに見合った動作をする(音を出す)わけだ．

レジスタという言葉を覚えたところで，さっきのSOUND命令の指定を，すこし専門家ふうに書き直してみると，こんな具合になる．

SOUND 〔レジスタ番号〕，〔データ〕

郵便箱のたとえをしつこく持ち出せば，SOUND命令であたえる2つの数字のうち，はじめのひとつは「手紙のあて先」，あとのひとつは「手紙の内容」と言えるだろう．

ここで，本文とは何の関係もないが，「SOUND」とタイプする練習だと思って，次のプログラムを入力してみよう．いちいち「SOUND」と入力するのすら面倒だという横着な人は，いったん

KEY 1, "SOUND" RETURN

とだけ，行番号をつけずにタイプしておけば，以後は F1 のキーを押すだけで，「SOUND」という文字が画面に出てくれる．

```
10 SOUND 7,56
20 SOUND 8,15
30 SOUND 9,15
40 SOUND 10,15
50 SOUND 0,RND(1)*255
60 SOUND 2,RND(1)*255
70 SOUND 4,RND(1)*255
80 GOTO 50
```

RUNしてみると，なんだか得体の知れない音が聞こえただろう．PSGには，こんな奇怪な音も出せるのである．

では，次に各レジスタにわたすデータの意味について，順番に説明していこう．

音源を選ぶ 3-3

まず，各チャンネルの音源を選ぶ話から．音源の選択は，レジスタの７番で行う．この７番のレジスタというのは，このあとで説明する他のレジスタとは，少々おもむきが変わっている．というのは，データのあたえ方が，ちょっとややこしいのだ．

コンピュータの内部では，すべての情報が「０」と「１」の２進数の符号で表わされている．これは読者もご存じだろう．この符号のことを専門用語ではビットと呼ぶ．

では，レジスタ７の内部をのぞいてみると，**図３－３**のように８つのビットがはいるようになっている．

レジスタ７の場合には，このひとつひとつのビットが，それぞれ各チャンネルの音源のON／OFFを決めるスイッチの役目をはたしている．

図３-３



・各ビットが音源に対するスイッチになっている.



右端から，0，1，2番のビットがチャンネルA，B，Cに対するトーン発振器のスイッチで，3，4，5番のビットがノイズ発振器のスイッチだ．スイッチは，ビットが「0」のときにはONになり，ビットが「1」のときにはOFFになる．残った2つのビット，6番と7番は，とりあえず使わないので無視しておこう．

なんだか，わけがわからなくなってきたかもしれない．具体的な例をいくつか見ながら考えてみよう．

まず，チャンネルAからトーンを出す場合．このときには，レジスタ7の中の各ビットが，次の形になるように，データを送りこんでやればよい．

あたえるデータの形式は，①のように，そのまま2進数の列のあたまに「&B」をつけたものでもいいし，②のように，10進数に直したものでもよい．

	B7	B6	B5	B4	B3	B2	B1	B0
音の種類			ノイズ			トーン		
チャンネル			C	B	A	C	B	A
ビット	0	0	1	1	1	1	1	0 (ON)

```
① SOUND 7,&B00111110
② SOUND 7,62
```

次に，チャンネルBから，トーンとノイズの両方を同時に出す場合．これは下のようになる．

	B7	B6	B5	B4	B3	B2	B1	B0
音の種類			ノイズ			トーン		
チャンネル			C	B	A	C	B	A
ビット	0	0	1	0 (ON)	1	1	0 (ON)	1

```
① SOUND 7,&B00101101
② SOUND 7,45
```

もうひとつ，3つのチャンネルすべてのトーンをONにして3重和音を出した上に，さらにチャンネルBからノイズをかぶせる場合．このときには，レジスタ7番のビット0，1，2とビット4を「0」にしてやる．

	B7	B6	B5	B4	B3	B2	B1	B0
音の種類			ノイズ			トーン		
チャンネル			C	B	A	C	B	A
ビット	0	0	1	0	1	0	0	0

ON

① SOUND 7,&B00101000
② SOUND 7,40

このように，レジスタ7番を使って，各チャンネルごとにトーンを出すか，ノイズを出すか，それとも両方かを自由に選択できるわけだ．

あたえるデータの形式は，2進法でも，10進法でも，どちらでもかまわない．2進数表記のほうは，一目でどのチャンネルのどの音源がONになっているかわかるし，あとで変更するときも楽だ．はじめのうちは，2進表記を使うとよいだろう．

けれども，いったん変数などに読みこんだデータは，再びプリントなどさせると10進表記に変わってしまうので，後者も覚えておこう．

図3－4に，10進表記の値を求める便利な方法を示しておく．

（10進換算早見表）　　　　　　　　　　　　　　　　　　　　**図3-4**

音の種類	ノイズ			トーン		
チャンネル	C	B	A	C	B	A
63から引く値	32	16	8	4	2	1

・「ON」にしたいスイッチに対応する数を左の早見表から求めて，その合計を「63」から引く．
　たとえば「チャンネルB，ノイズ/ON，トーン/ON」ならば，63－(16＋2)＝45が求める値．

3-4 音の高さを決める

次に音の高さの話に移ろう．

0番から5番までの6つは，各チャンネルの音の高さを決めるレジスタだ．1つのチャンネルの音の高さを決めるには，2つのレジスタをペアで使う必要がある．チャンネルAはレジスタの0番と1番を，チャンネルBはレジスタの2番と3番を，チャンネルCはレジスタの4番と5番を，それぞれ使う．

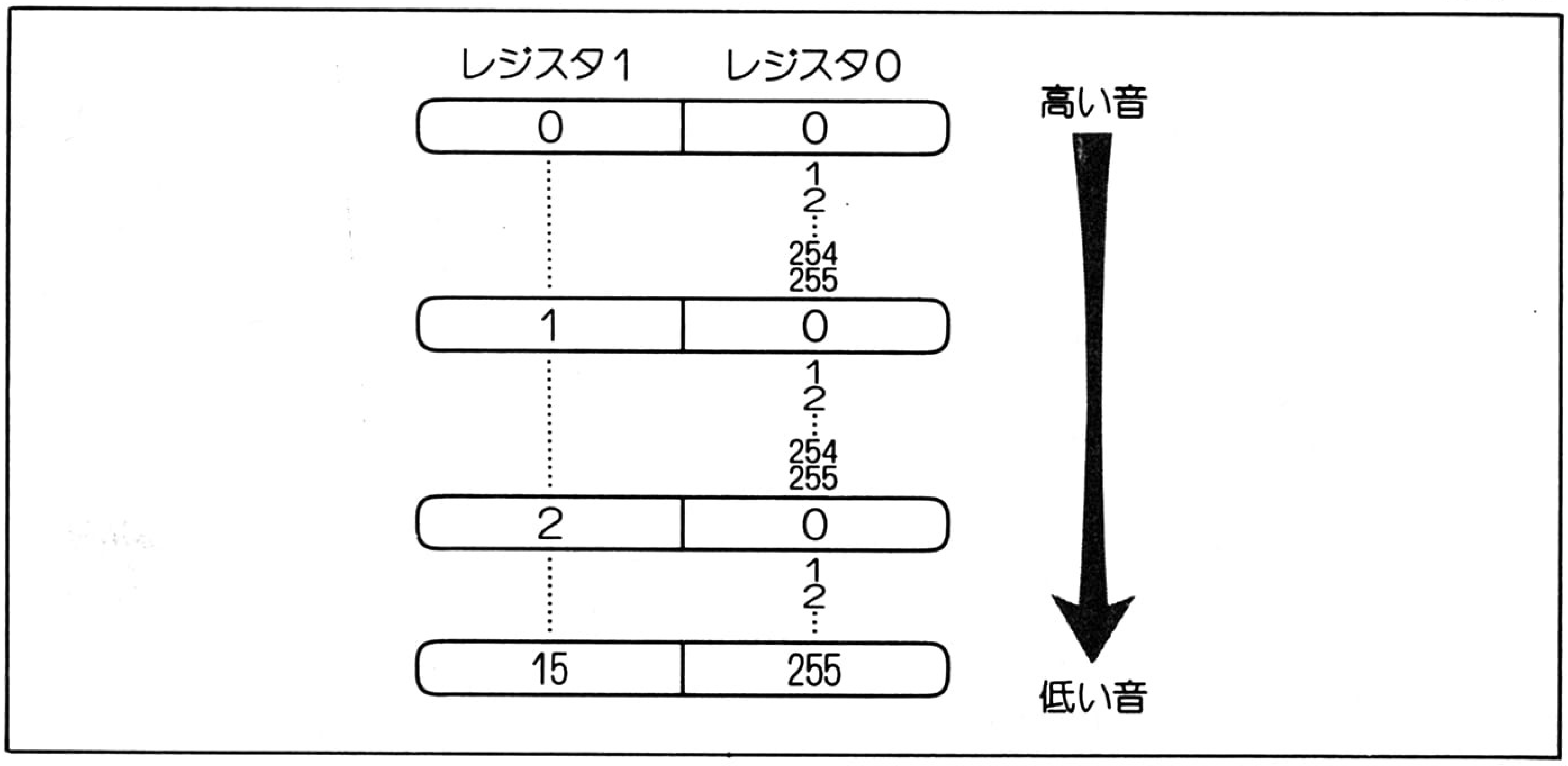
どうして2つも必要なのかと聞かれても，そういうふうに作られているから，としか答えようがない．チャンネルAを例にとって，目指す高さの音を得るにはどのような数値をデータとしてあたえればよいか，それを見ることにしよう．

まず，レジスタの0番．これには，0から255までの数値がはいる．
次に，レジスタの1番．これには，0から15までの数値がはいる．

で，両者の関係はというと，**図3－5**のように2つのレジスタがならんでいて，レジスタの0番が「255」を迎えていっぱいになるたびに，レジスタの1番が，「1」繰り上がる，と考えてもらえばよい．

言いかえると，PSGで出せる音の高さの範囲は，おおまかに16段階に区切られていて(レジスタ1の値)，その中がさらに256の細かいきざみ(レジスタ0の値)になっているということだ．

図3－5



これを，レジスタ1がおおざっぱなチューニング，レジスタ0がファイン・チューニング(微調整)と考えてもらってもよい．

次のリストは，PSGが出せる音を上から下まですべての範囲で出してみるプログラムだ．ほとんど聞こえないような，耳障りなカン高い音からはじまって，だんだんふつうの音になり，最後には，超低音となってむしろプツプツという音になってしまうのがわかるだろう．

```
10 SOUND 7,62
20 SOUND 8,15
30 FOR I=0 TO 15
40   FOR J=0 TO 255
50     SOUND 1,I
60     SOUND 0,J
70   NEXT J
80 NEXT I
90 SOUND 8,0
```



注意してほしいのは，レジスタにあたえる数値が大きいときほど「低い」音が出るという点だ．これはちょうど，音の高さと周波数の高さの関係とは，逆になっている．だいたいパソコンというやつは，グラフィック座標のY軸といい，どうもヘソ曲がりなところが多いのだ．

さて，ここで，「どうしてもアタシャ440Hzの音がほしい！」というガンコな人のために，周波数からパソコンにあたえるべきデータを算出する式を紹介しておこう．まず，次の式のfのところに，ほしい周波数の値を代入して，nの値を求める．

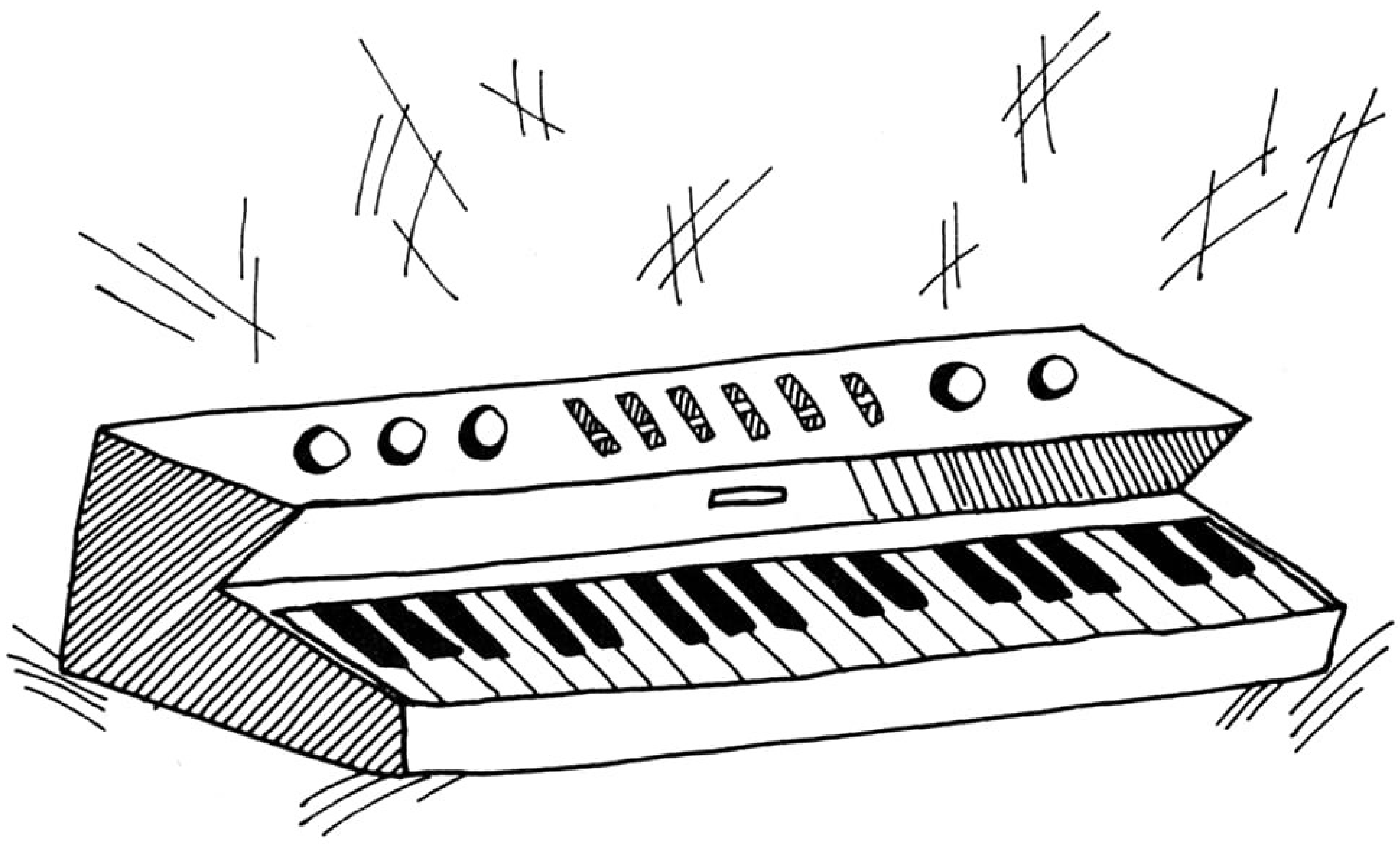
$$n = \frac{1789800}{16 \times f}$$ ［fはほしい周波数の値(Hz)］

次にこのnを256で割る．割ったときの商の値が，レジスタ1のデータ，余りが，レジスタ0のデータとなる．なお，上の式に出てきた，「1789800」というやたらめったら大きい数字は，「クロック周波数」といって，PSGがほかの周波数を作るときに，基準にする値だ．

こんな面倒くさい計算は，それこそパソコンにでもやらせたらよろしい，というわけで，次に，目指す周波数に対応するデータの値をはじき出すプログラムを載せておく．RUNしたのち，質問にこたえて適当な周波数を入力してやると，2つのレジスタの値が求められる．

レジスタ(H)というのが，各チャンネルのおおまかなきざみを決める1，3，5番のレジスタにあたえるべきデータ，レジスタ(L)というのが，細かいほうの0，2，4番レジスタにあたえるべきデータである．ちなみに，周波数＝440とすると，レジスタ(H)＝0，レジスタ(L)＝254となる．

```
100 CLEAR:FC=1789800#
110 INPUT "ｼｭｳﾊｽｳ (Hz)= ";F
120 TP=INT(FC/(16*F))
130 RL=TP MOD 256
140 RH=INT(TP/256)
150 PRINT "ﾚｼﾞｽﾀ (L)=";RL
160 PRINT "ﾚｼﾞｽﾀ (H)=";RH
170 GOTO 100
```



ノイズの種類を決める 3-5

さて，ノイズについても，レジスタの６番にデータを送りこんでやることで，多少のバラエティを得ることができる．

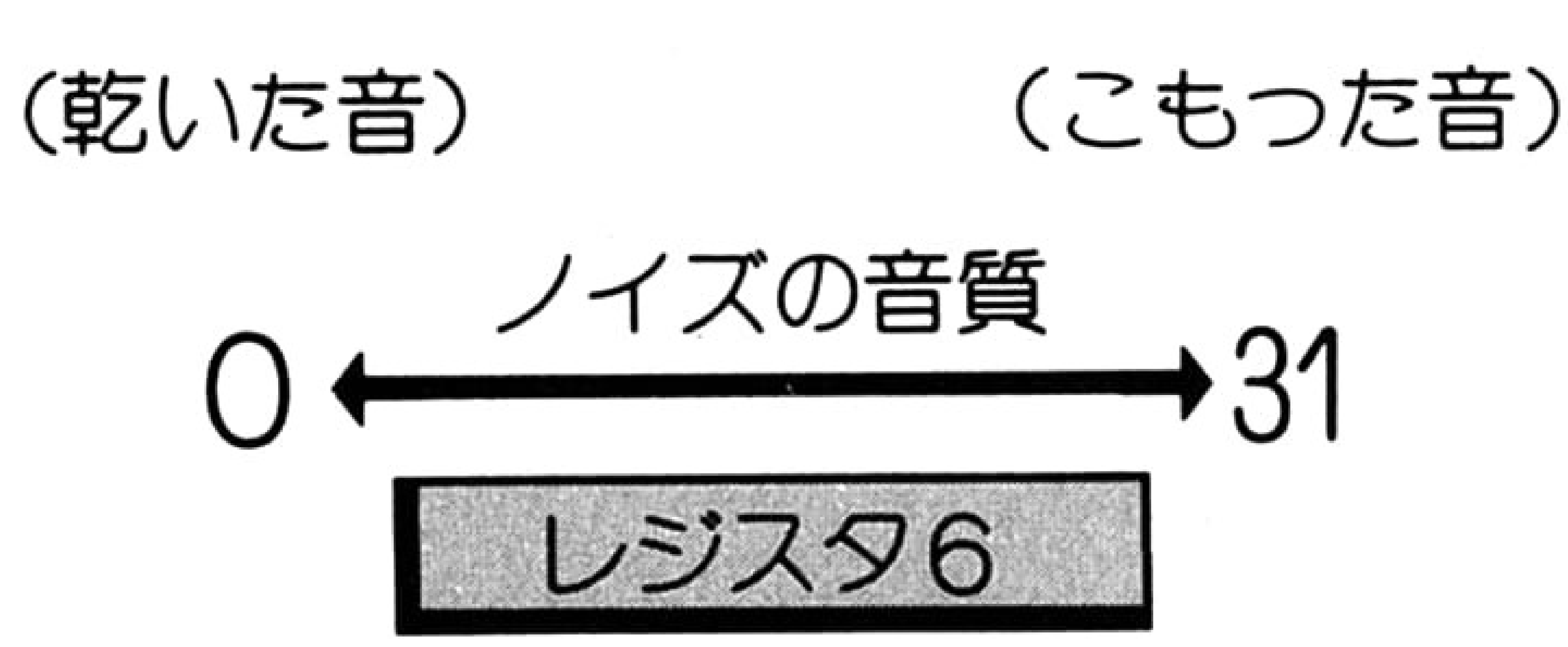
次のプログラムを実行してみよう．

```
10 SOUND 7,55
20 SOUND 8,15
30 FOR I=0 TO 31
40 SOUND 6,I
50 I$=INPUT$(1)
60 PRINT "ﾚｼﾞｽﾀ6=";I
70 NEXT I
```

RUN させてみると，まず，「サー」という乾いた感じのノイズが聞こえてくるはずだ．

次に，どれでもいいから，なにかキーを押してみよう．どんどん続けてキーを押していくと，しだいに「グォー」というこもった感じの音に変わっていくのがわかると思う．

このように，レジスタの６番にあたえる数値によって，ノイズの雰囲気を変えてやることができる．もう少しくわしくいうと，レジスタ６は，ノイズの周波数を設定する働きを持っている．



データとしてあたえられる数値の範囲は，「０」から「31」までで，数値の小さいほうが，軽い感じのノイズになる．これをうまく組み合わせてやれば，たとえば，ロケットが近づいてくるときの音などに利用できる．

3-6 音の大きさを決める

「音色」や「高さ」が決まったら，もうひとつ，音の「大きさ」を設定してやろう．PSGからなにか音を出すためには，最低限この3つを設定する必要がある．

各チャンネルの音の大きさは，レジスタの8番，9番，10番の3つによって決められる．チャンネルAがレジスタ8，チャンネルBがレジスタ9，チャンネルCがレジスタ10にそれぞれ対応している．

音の大きさの設定は，音の高さなどのときにくらべれば，ずっと簡単だ．

(無音)　　　　　　　　(最大)

0 ←――――――→ 15

レジスタ8	チャンネルA
レジスタ9	チャンネルB
レジスタ10	チャンネルC

PSGの各チャンネルの音量は，「0」から「15」までの数値として指定してやる．数値が大きいほど，音も大きくなり，「15」のときが音量最大，「0」のときが最小(つまり何も聞こえない状態)になる．たとえば，チャンネルAの音量を最大にするには，

```
SOUND 8, 15
```

と命令してやればよい．

もちろん，せっかく音の大きさを決めても，アンプのほうの音量(テレビなどのボリューム)を上げておかないと，音は聞こえない．

それでは，レジスタの設定方法をある程度覚えたところで，具体的に簡単な音を出してみることにしよう．

```
10 SOUND 7,60
20 SOUND 0,254:SOUND 1,0
30 SOUND 2,127:SOUND 3,0
40 SOUND 8,15
50 SOUND 9,15
```

このプログラムは，チャンネルAから440Hzの音を，チャンネルBから880Hzの音を2ついっしょに出すものだ．ちょうど1オクターブ離れた音が聞こえると思う．

ちなみに，440Hzの音とは，時報のあたまの「ポッ・ポッ・ポッ」という音と同じ高さで，オタマジャクシで言うと，ト音記号のちょうど真中あたりの「ラ」の音ということになる．

2つの音は，倍音関係になっているので，ちがいを聞き分けるのが少しむずかしいかも知れない．そこで，さらに次のようなプログラムを入力してみよう．今度はA，B，C，3つのチャンネルとも音量レジスタを15にして，3つ一度にトーンを出している．

```
10 SOUND 7,56
20 SOUND 0,254:SOUND 1,0
30 SOUND 2,127:SOUND 3,0
40 SOUND 4,63:SOUND 5,0
50 SOUND 8,15
60 SOUND 9,15
70 SOUND 10,15
```

よく聞くと，もう1オクターブ高い音がつけ加わったのがわかるはずだ．

また，プログラムをよく見ると，レジスタ2の値はレジスタ0の値のちょうど½，レジスタ4の値はほぼ¼になっていることがわかる．このように，音の高さのレジスタ値を半分にすると，ちょうど1オクターブ上の音が出る．

3-7「エンベロープ」を決める

さて，最後にエンベロープの使い方について．ここのところは，PSG による音作りの，いわば「キモ」にあたるので，しっかりと理解しておいてほしい．

これまで聞いてきた PSG の音は，いずれも「ピー」と鳴りっぱなしが多かった．けれども，これでは，現実の音とは程遠い．

ふつう，ボクらが身のまわりで聞く音は，出だしの音量がいちばん大きく，それからだんだんと消え去っていく，という特徴を持っている．この時間的な音量のちがいによって，「ポン」とか，「ポワーン」といった音「らしさ」が生まれてくるのだ．

本物のシンセサイザーでは，エンベロープについても自由度が高く，立ち上がりの強さからどのくらい余韻を残すかということまで，細かく指定できる仕組みになっている．けれども，MSX内蔵のPSGだって負けてはいない．

PSG には，こうした時間的な音量変化の型(エンベロープ・パターン)が，全部で 8 つ登録されている．あらかじめ作っておいた音に，このエンベロープの型をあてはめてやることで，本物らしい音に近づけることができる．

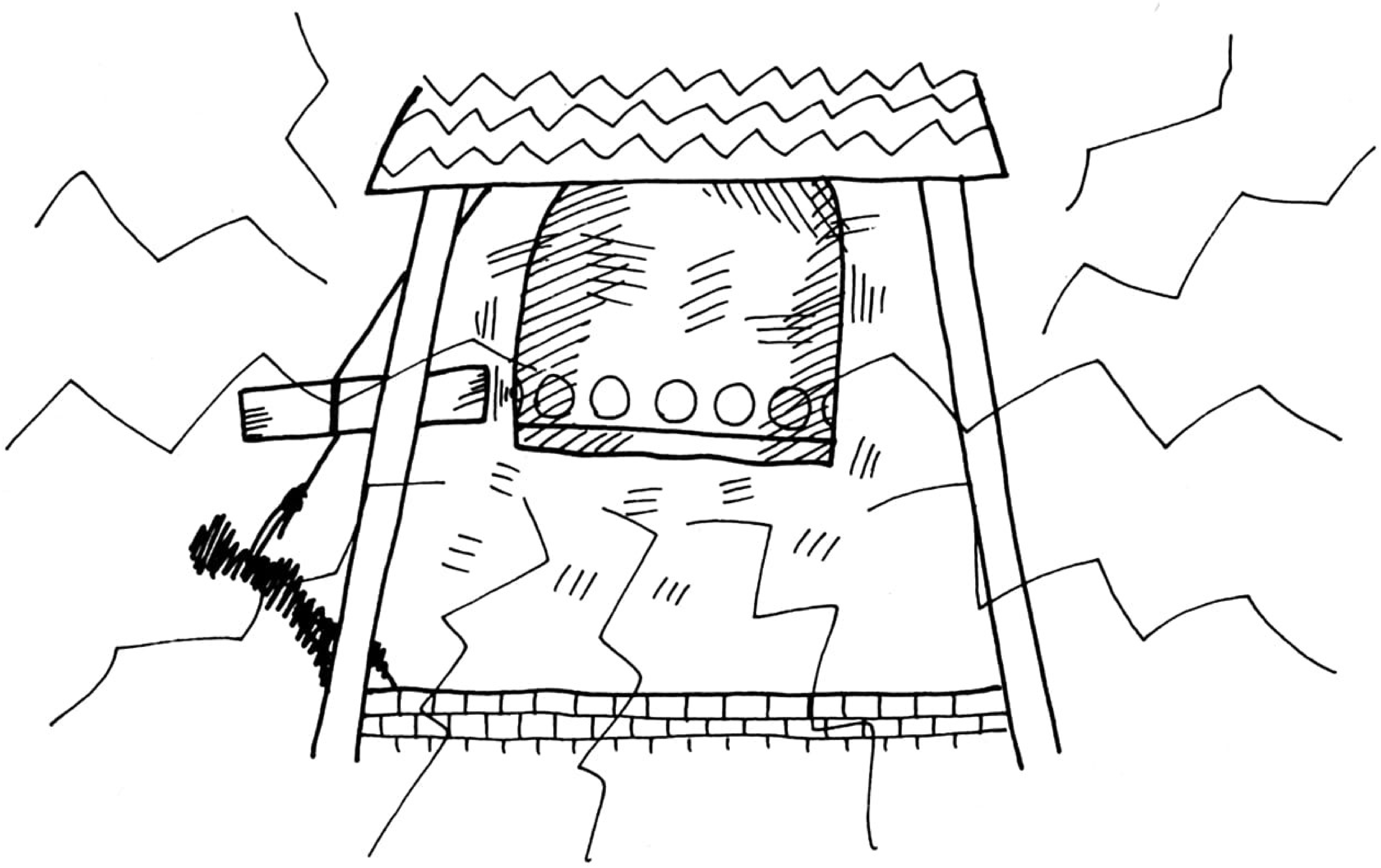


図3－6は，PSGの持つ8つのエンベロープ・パターンを，よこ軸に時間を，たて軸に音量をとって，書き表わしたものだ．

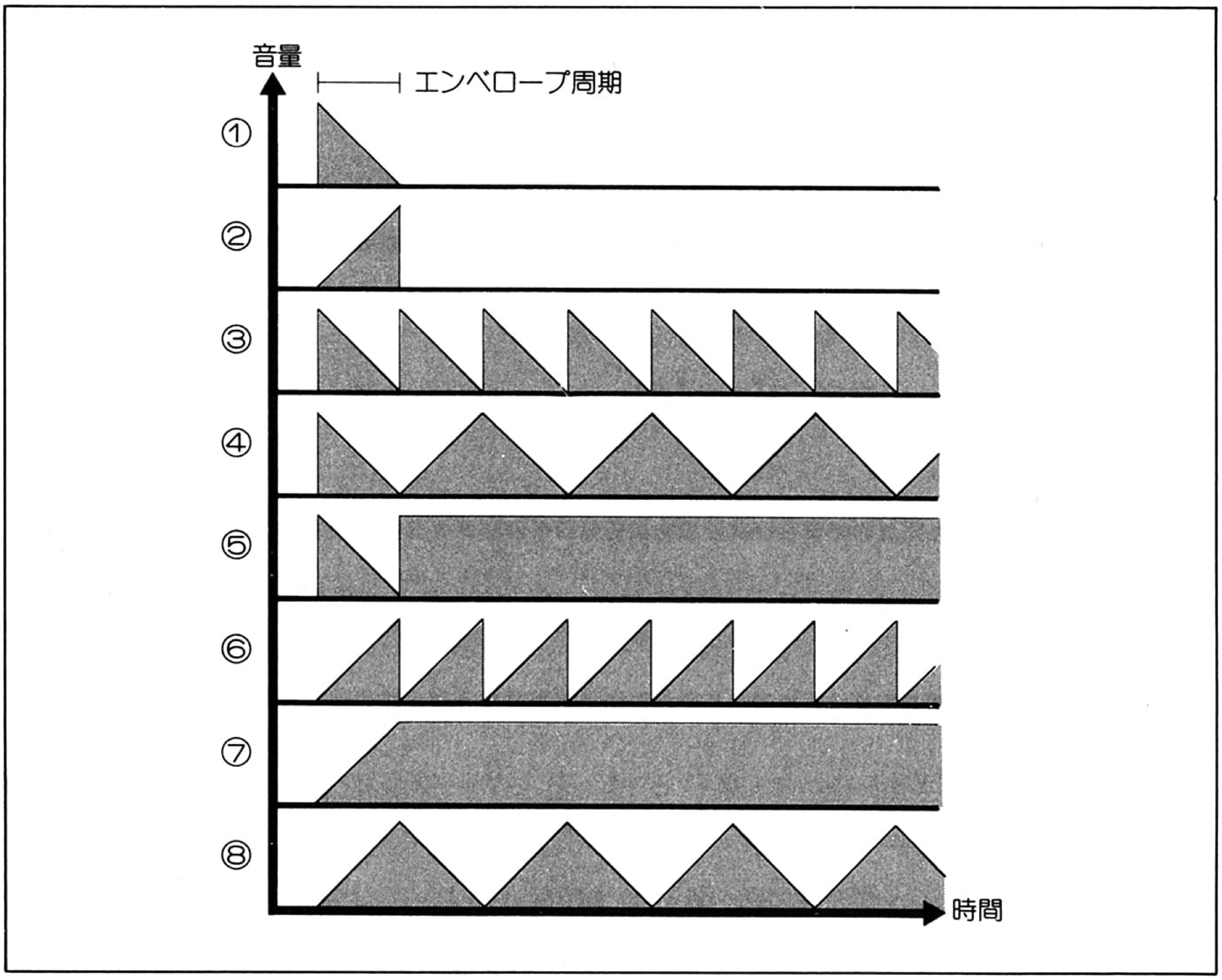
たとえば，①のパターンは，いちばんふつうの音に近く，「ポン」と一回音を出して消えてしまうものだ．このような一回限りの音のパターンを，俗称「一発波形」などと呼ぶ．⑧の場合には，波の音のように，大きくなったり小さくなったりが，何度も繰り返される．

音が出てから消え去るまでの時間(図3－6で言えば，山ひとつぶんの時間)を，エンベロープの「周期」という．

PSGは，この周期も変えることができる．①のパターンの場合，周期の長さを調節することで，歯切れのいい短い音から，なが～く続く音まで出すことができる．

いっぽう，③のように，連続したパターンでは，周期を短くすると，「ピリピリピリピリ」というブザーのような音になる．

図3-6



エンベロープの設定には，レジスタの11番から13番までを使うが，くわしい設定のしかたはあとにまわして，とりあえず，エンベロープによる音のちがいを聞いてみることにしよう．

```
10 SOUND 7,62
20 SOUND 1,0:SOUND 0,254
30 SOUND 8,16
40 SOUND 13,9
50 SOUND 12,60:SOUND 11,0
```

上のリストをRUNさせてみると，「ポーン」とチャイムのような音が出たと思う．これが，①のエンベロープ・パターンの音だ．

次に，リストの40行目を，

```
40 SOUND 13,8
```

に変えてみる．RUNすると，「ポーン・ポーン・ポーン」とさっきと同じ音が，何度も続けて出る．これが，③のパターンだ．

さらに，40行目を，

```
40 SOUND 13,12
```

とすると，今度は「スーゥパ・スーゥパ」と，テープを逆まわしにしたような音になる．これはパターンの⑥だ．

以上3つの音は，いずれも同じ周波数440Hzのトーンだが，音源をノイズに変えてみると，もっといろいろなバラエティが出せる．

```
10 SOUND 7,55
20 SOUND 6,15
30 SOUND 8,16
40 SOUND 13,12
50 SOUND 12,60:SOUND 11,0
```

どうだろう，なんとなくインベーダーがつぎつぎと迫ってくるような感じに聞こえないかな？

さらに，周期のほうも変えてみよう．上のリストの50行目を書きかえる．

```
50 SOUND 12,2:SOUND 11,0
```

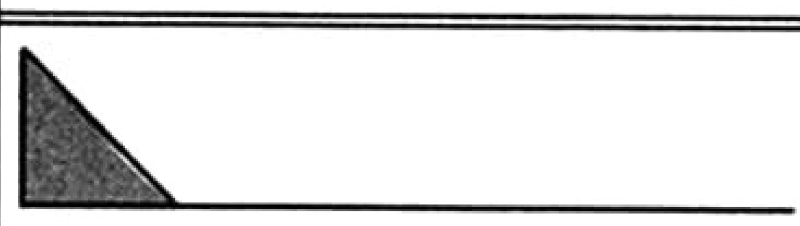
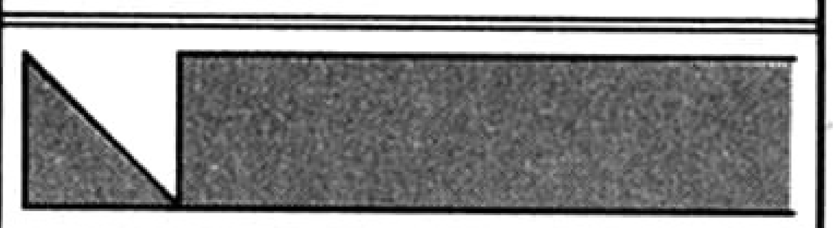
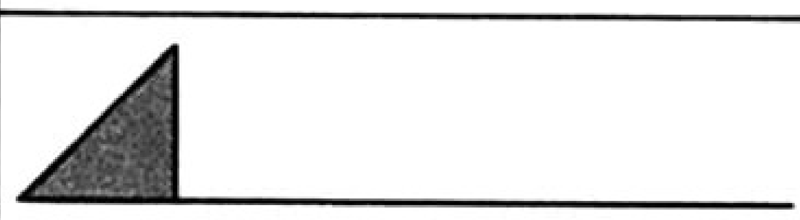
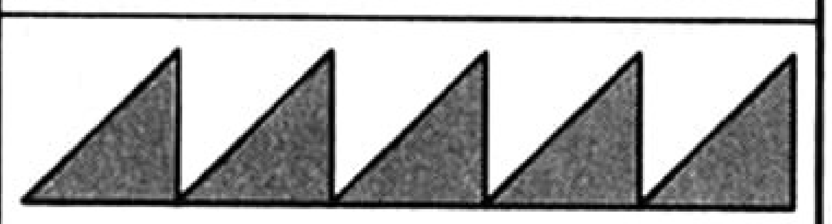
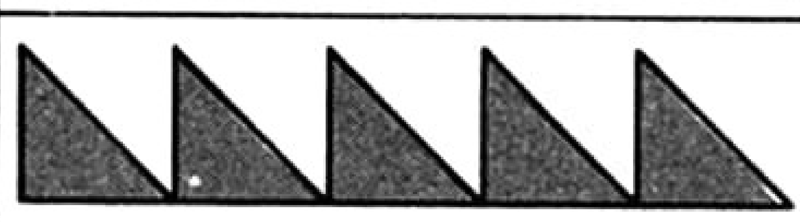
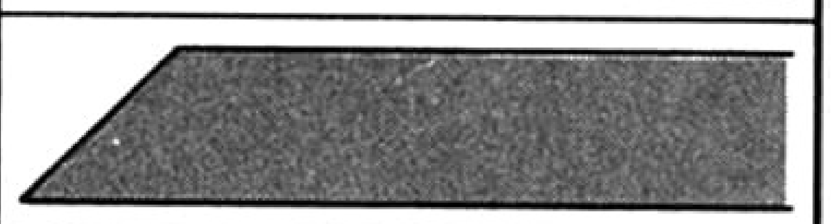
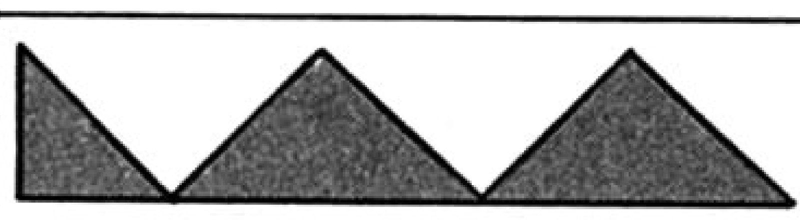
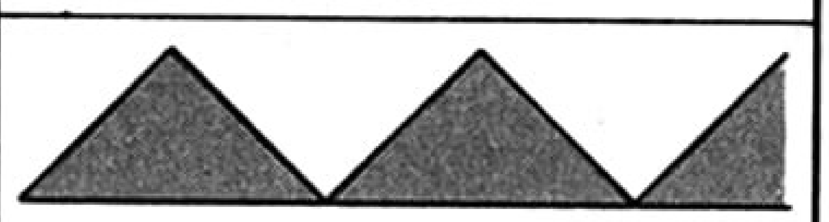
「スパスパスパスパ」と，すごい速さで音が出た．光線銃なんかの音(？)に使えそうだ．

このように，エンベロープしだいで，ずいぶんいろんな感じが出せるのだ．

それでは，レジスタ11，12，13と，あたえるデータの意味について，説明しよう．

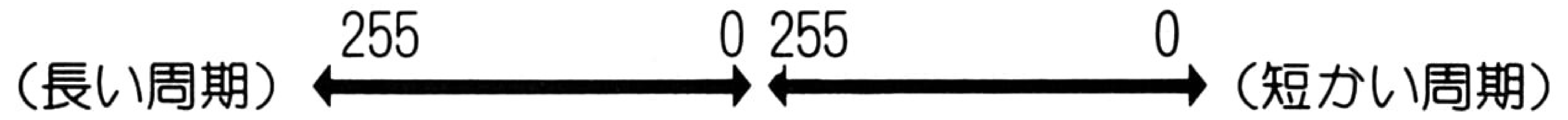
まず，レジスタの13番．これは，エンベロープの波のパターンを選ぶ働きをしていて，「0」から「15」までの数値を入れる．といっても，エンベロープのパターンは8つしかないので，0～3と9のときはパターンの①が，4～7と15のときはパターンの②が選ばれる．そのほかの場合は，**図3－7**のとおりだ．

図3-7

レジスタ13の値	エンベロープ・パターン	レジスタ13の値	エンベロープ・パターン
0～3 および9		11	
4～7 および15		12	
8		13	
10		14	

次に，レジスタの11と12だが，この２つをペアで使って，エンベロープの周期を決める．両方とも，「０」から「255」までのデータが有効で，〔レジスタ12の内容〕×256＋〔レジスタ11の内容〕が周期としてセットされる．

つまり，レジスタ12が大まかな目盛り，レジスタ11が細かい目盛りになっていると思えばよい．大きい数値を入れるほど，周期は長くなる．



大まかな調節	細かい調節
レジスタ12	レジスタ11

参考までに，望みの長さの周期を得るには，どのくらいの数値をあたえればいいのか，ということを見ておこう．

$$n = \frac{1789800}{256/T}$$ ［Tはほしい周期の値(秒)］

上の式のＴに，ほしい周期の長さ(秒)を代入して，ｎの値を求める．このｎを256で割った商がレジスタ12の値，余りがレジスタ11の値だ．

ちなみに，レジスタ11，12ともに255に設定したとき(いちばん長くしたとき)には，周期はだいたい4.2秒くらいになる．

さて，こうしてパターンと周期を決めたら，どのチャンネルにエンベロープをかけるかということを指定する．この指定は，音量設定のときに使ったレジスタ８，９，10で行う．

さきほど「音の大きさ」を決めるレジスタには，「０」から「15」までの数値がはいると述べたが，じつはこのレジスタは，各チャンネルにエンベロープをかけるか，かけないかというスイッチの役目もかねている．これらのレジスタに「16」をセットすると，対応するチャンネルがエンベロープ・モードとなる．

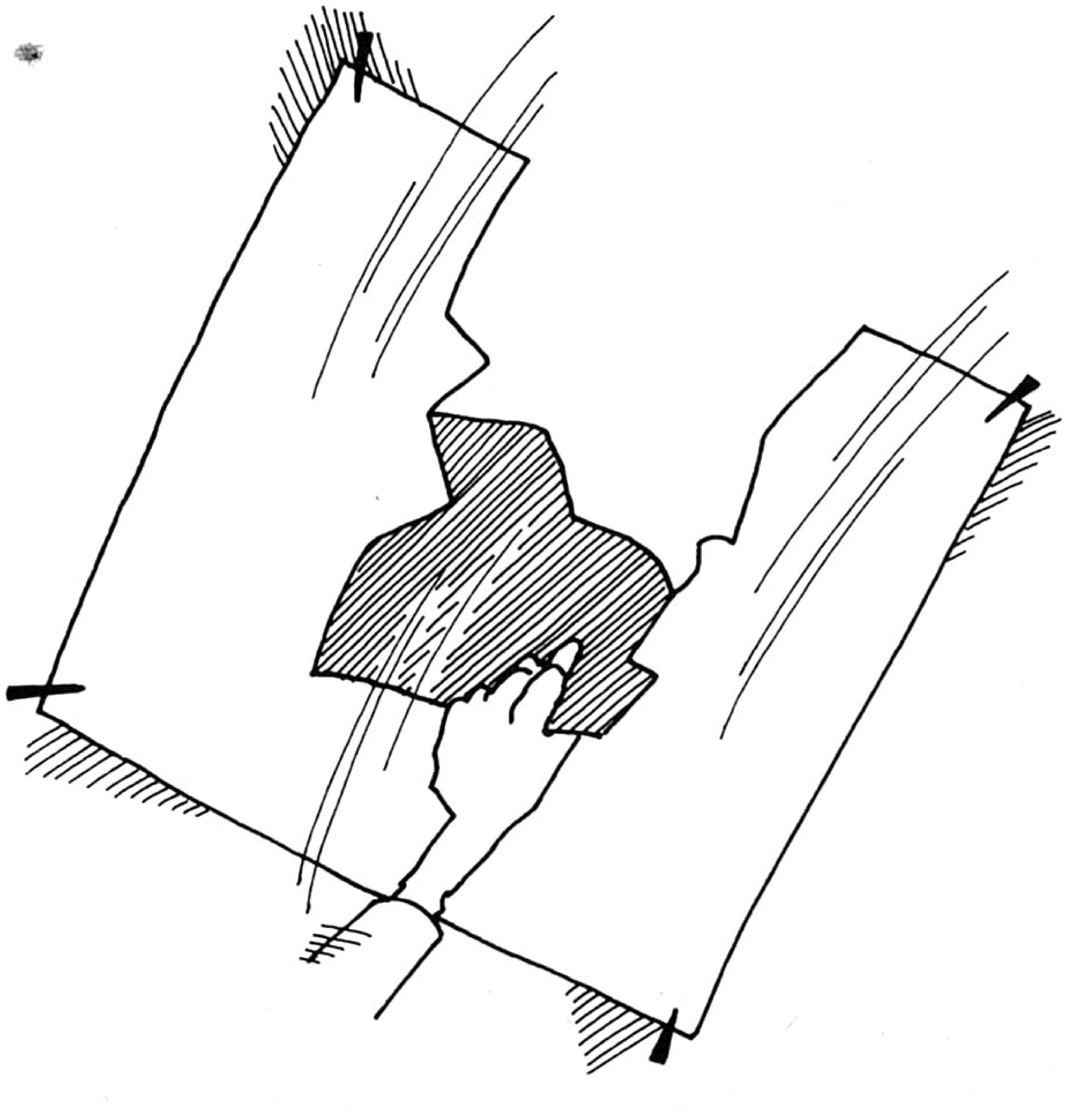
レジスタ	チャンネル	音量制御	エンベロープON
8	A	0~15	16
9	B	0~15	16
10	C	0~15	16

たとえば，チャンネルAにエンベロープをかけるには，

```
SOUND 8, 16
```

と命令する．

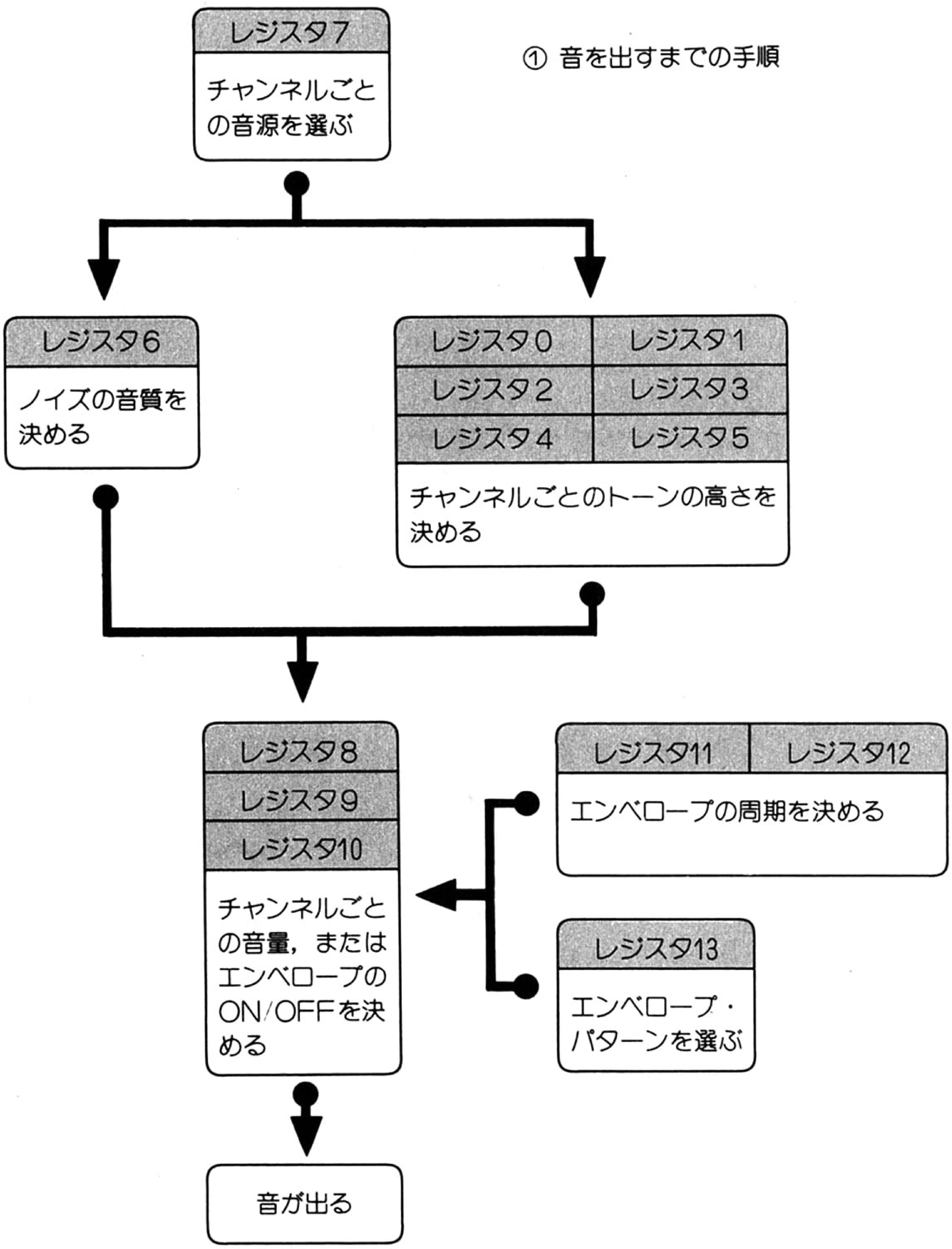
これでわかるように，エンベロープは各チャンネルごとに使うか使わないかの指示ができるだけで，AとBにちがう型を選ぶことはできない．本格的な音作りには，ちょっと物足りない気もするが，まあしかたのないところだろう．



まとめ

さて，これでPSGのレジスタと，SOUND命令についての説明はすべて終わった．なんだかたくさんあったが，いっぺんに覚えようとする必要もない．使っていくうちに，だんだんと慣れてくるだろう．

最後に，SOUND命令の使い方メモを載せておくので，役に立ててほしい．



② 14コのレジスタの働き

レジスタ番号	レジスタの働き	データの範囲	データの意味
0	チャンネルAの高さ 微調整	0～255	数値が大きいほど音は低くなる.
1	チャンネルAの高さ 大まかな調整	0～15	
2	チャンネルBの高さ 微調整	0～255	数値が大きいほど音は低くなる.
3	チャンネルBの高さ 大まかな調整	0～15	
4	チャンネルCの高さ 微調整	0～255	数値が大きいほど音は低くなる.
5	チャンネルCの高さ 大まかな調整	0～15	
6	ノイズの音質	0～31	数値が大きいほどこもった感じの音になる.
7	各チャンネルの音源の選択	0～63	0～5までのビットが，各チャンネルのトーン，ノイズのON/OFFスイッチとなっている.「0」でON，「1」でOFF.
8	チャンネルAの音の大きさ	0～16	「0」が最小(無音)，「15」が最大.「16」を選んだときは，エンベロープ・モードとなる.
9	チャンネルBの音の大きさ	0～16	
10	チャンネルCの音の大きさ	0～16	
11	エンベロープの周期 微調整	0～255	エンベロープの周期を決める. 数値が大きいほど周期は長くなる.
12	エンベロープの周期 大まかな調整	0～255	
13	エンベロープ・パターン	0～15	8種類のエンベロープ・パターンを選ぶ

BLOCK FOUR

4 効果音の作り方

――――サウンド・エディタによる音作り

この章では，PSGを使って望みの効果音を作るための，より実戦的なノウハウについて述べていく．

SOUND命令とPSGの14個のレジスタの働きについては，あらかた理解してもらえたと思うが，では，いったいどんな周波数のトーンやノイズに，どんなエンベロープをかけると，どういう音ができ上がるのだろうか．

このあたりのことは，なかなか一口で説明するのはむずかしく，あれこれと試してみて，体験的につかんでもらうのがいちばんだ．

じっさい，「ジェット機の音を作ろう」などと思っても，もともと現実の音がどんな周波数やエンベロープを持っているのか知らなければ，それを再現することはできない．といって，オシロスコープなどの分析器もなく，PSGという限られた機能のシンセしか持たないボクらにとっては，これはキツい作業だ．

むしろ，SOUND命令にあたえるデータの値をあれこれいじってみて，だんだん自分のイメージに合った音に近づけていくほうが，手っ取り早い．

ときには，思いもかけない非現実的な音ができ上がって，ビックリすることもあるだろう．ゲームなどの効果音は，かえってこんな音のほうが，雰囲気が出たりするものだ．



サウンド・エディタを作ろう 4-1

まず，ものを作るためには，道具が必要だ．PSG から出る音のテストを行うのに，いちいち SOUND 命令を書きかえていたのでは，たいへんに面倒くさい．そこで，本書では，効果音開発用のツール・プログラムを 2 つ紹介することにした．

ひとつは，14個のレジスタにそのままデータを書きこんでいくもので，PSG の各レジスタの働きを知るにはとてもよいだろう．

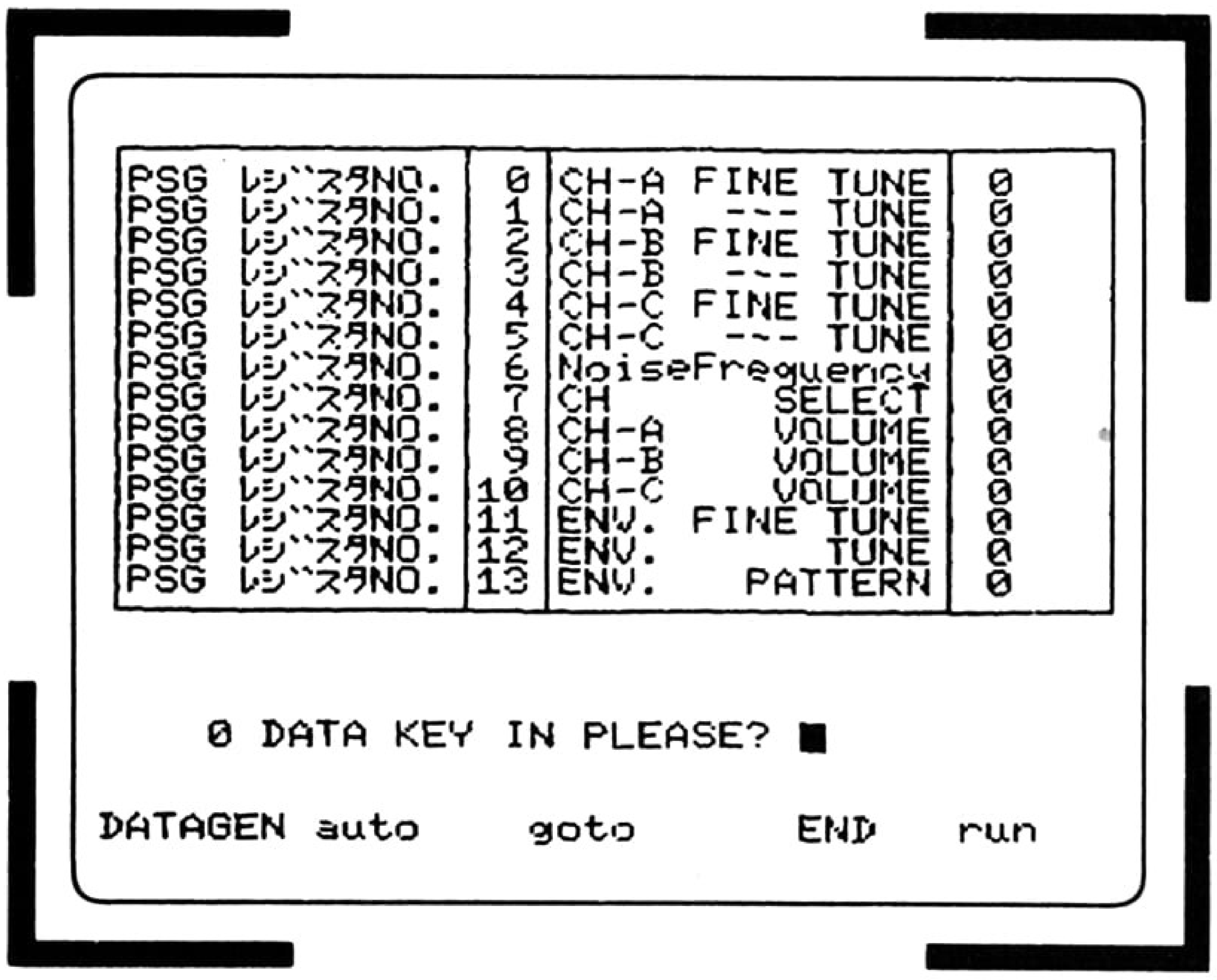
もうひとつは，音の高低やエンベロープの周期などをカーソル・キーを使ってスライドさせていくプログラムで，操作性の面では，こちらのほうが使いやすくなっている．

作り上げた音のデータは，どちらもデータ文の形で，保存することができる．

それでは，はじめのほうのツールの使い方から説明していくことにしよう．

42 ページのプログラム，「**サウンド・エディター 1**」を，間違えないように入力する．キー操作に慣れないひとには，ちょっとたいへんかもしれないが，あせらず 1 行ずつ確かめながら入力していこう．終わったら，テープにセーブする．以後，動作ミスなどが起きたら，プログラムをチェックし直す．

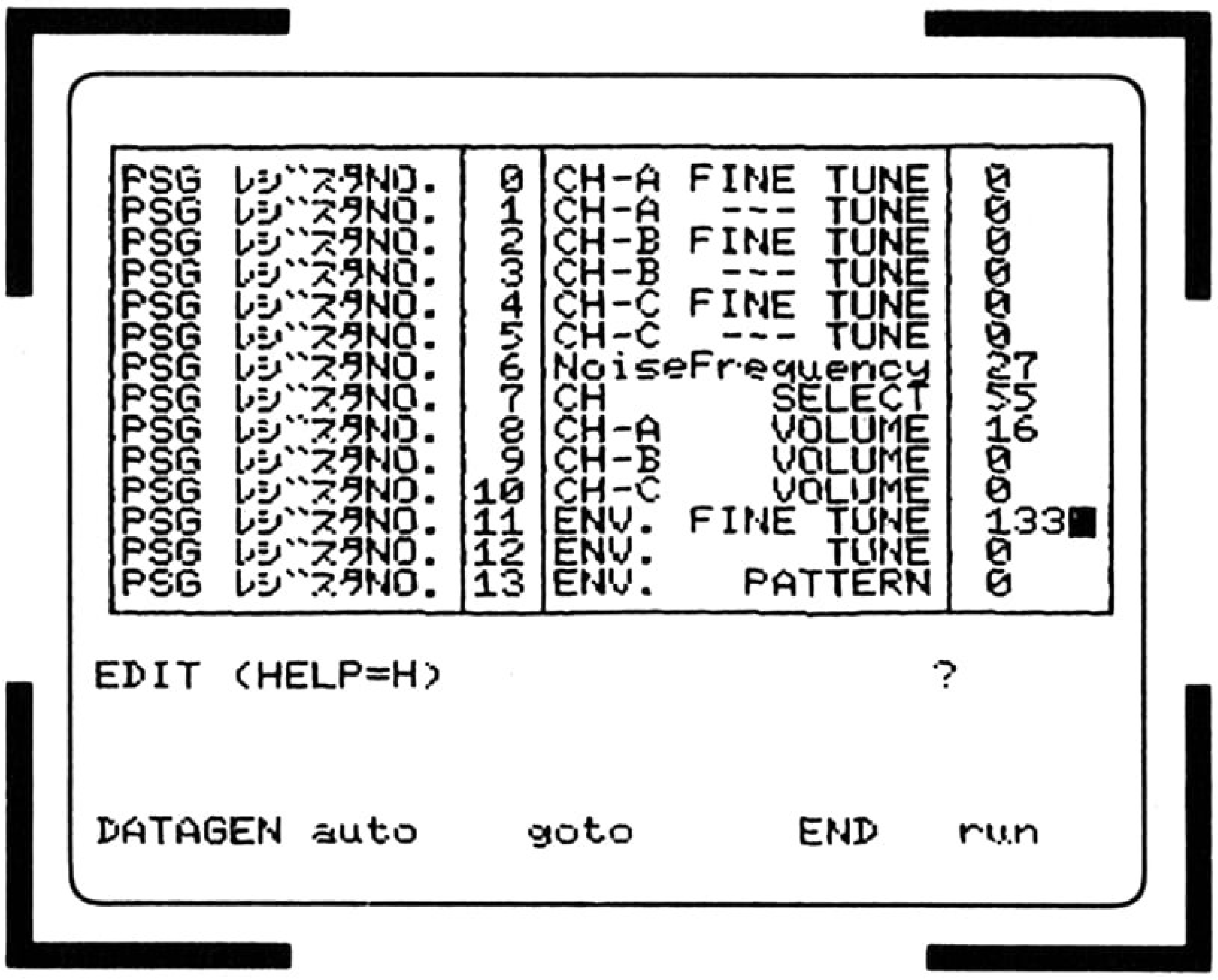
RUN させると，まず14個のレジスタの一覧が表示される．



このあと画面の下のほうに質問が出て，0 番から順番に，各レジスタの値をきいてくる．巻末に掲げたデータ集のように，すでに14個の値が用意してあるとき

には，ここでそのデータを入力していく．なにもない状態から音作りをはじめたいときには，ただ RETURN キーを押していけば，全レジスタに「0」がセットされる．

ひととおり入力が終わったら，次に「編集(エディット)」の作業にはいる．

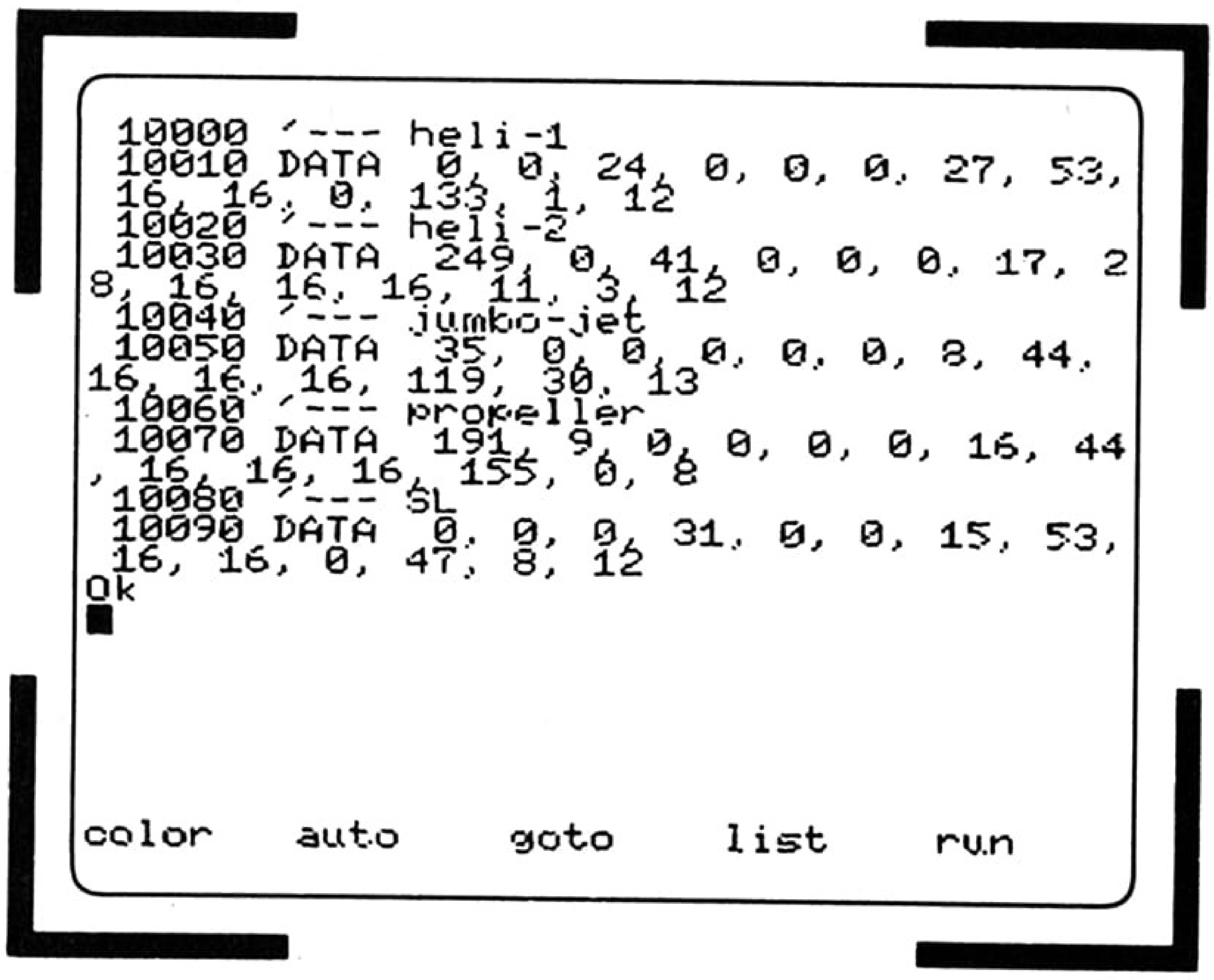


設定し直したいレジスタの数字のところにカーソルを動かしていって，新たな値を書きこみ，RETURN を押す．すると，自動的に PSG に対してデータが送りこまれ，音が変化するはずだ．

エンベロープの型によっては，一発しか音が出ない場合もあるので，こんなときには，カーソルを動かす前に RETURN だけを押すと，もういちど同じ音が聞ける．こうして，各レジスタの値を見ながら，音のほうは耳で確かめながら，いろいろな場合を試してみることができる．

満足のいく音ができ上がったら，F 1 キー，RETURN キーと続けて押そう．「DATA NAME ?」と質問が出るので，適当な名前を入力して，もういちど RETURN を押す．これで，レジスタ内のデータがワンセットになって，パソコンに記憶される．

記憶できるデータの数は 5 組までで，いっぱいになると自動的に「DATA FULL」と表示が出たのち，5 組のデータが DATA 文の形で表示される．これにカーソルを合わせて RETURN キーを押すことで，エディタ・プログラム本体に組みこんでしまえばよい．カセットにセーブするときにわずらわしければ，行番号10000以下だけを残して，本体を DELETE(削除)してやる．



ヘリコプターの音を作る 4-2

では，実際にこのツールを使って，「ヘリコプター」の音を作ってみることにしよう．

「サウンド・エディター1」をRUNさせたら，まずデータ入力の質問には，すべて RETURN でこたえておく．14個のレジスタに，「0」をセットし終わったら，「ヘリ」の音作りにはいる．

「ヘリ」の音は，「パタパタ」という連続音なので，エンベロープには「12」の右肩上がりのノコギリ型をえらぶ．もちろん，音源はノイズだ．

必要なレジスタの設定値は，次のとおりだ．カーソルを画面のデータのところまで動かして，値を書き直したのち，一回ごとに RETURN キーを押す．

レジスタ 7——— 55(チャンネルA/ノイズON)
レジスタ 6——— 27(ノイズの周波数設定)
レジスタ 8——— 16(チャンネルA/エンベロープON)
レジスタ11———133(エンベロープ周期の設定)
レジスタ12——— 1(エンベロープ周期の設定)
レジスタ13——— 12(エンベロープ・パターンの設定)

どうだい，上の５つの値を入れてやると，「パタパタパタパタ」と，なんだかヘリコプターみたいな音がしてきただろう？

なんかちょっと気ぜわしい感じがする人は，レジスタ12のデータを「２」に変えてみよう．こんどは，少しゆっくりとしたローターの回転音に聞こえるはずだ．いろいろいじってみて，気に入った速さを選ぼう．

さて，このままでも，けっこうヘリコプターらしいのだが，どうも田舎の耕運機みたいだという気がしないでもない．そこで，チャンネルBからうんと高い音を出して，ジェット・エンジンの感じを加えてやる．変更点は，次のとおりだ．

レジスタ 7——— 53(チャンネルA/ノイズON,
　　　　　　　　　チャンネルB/トーンON)
レジスタ 2——— 24(チャンネルBの周波数設定)
レジスタ 9——— 16(チャンネルB/エンベロープON)

これで，最新型のジェット・ヘリみたいな気分が出てきた．

このように，ちょっとレジスタの値を変えるだけで，ずいぶんちがった感じの音が出せる．本書では，オリジナルの音作りのためのガイドとして，巻末にたくさんのサンプル・データを載せておいた．これらのデータをもとに，どんどんおもしろい音作りにチャレンジしてみてほしい．

4-3 本格的サウンド・エディタの紹介

次に紹介するツール「**サウンド・エディター２**」(45ページのリスト）は，サウンドの編集をより簡単な操作で行えるように考えたものだ．

このツールでは，トーンの高低やエンベロープの周期をいちいち数値で入力してやらなくても，カーソル・キーによって自由に上下させることができる．これで，ちょっとしたシンセ演奏家の気分が味わえるはずだ．

また，でき上がった音をPLAY文でも利用できるように，ミュージック・マクロ言語とのデータ値の変換を行えるような配慮もしてある．

ノイズについては，ON／OFFの切りかえしかできないが，これは，ツールの

操作があまりにもゴチャゴチャと複雑になるのを避けたものだ．プログラムは，すべてBASICで書かれているので，興味のある人は，ノイズの周波数も変えられるように，改良を計ってみるとよいだろう．

では，このツールの取り扱い説明に移ろう．各キーの働きについて説明する．なお，各コマンドの入力は，すべて小文字で行う．

▶ ↑ ↓ ← →（カーソルキー）

画面に出ているとおり，上下は音の高さを，左右は音の大きさを変化させる．また，エンベロープがかかっているときには，左右はエンベロープの周期を変える．これらの表示は画面の下のほうに数値で表わされるが，この値は2つのレジスタをつなげたものなので，注意しよう（音の高さのとき，最大値4095，エンベロープ周期のとき，最大値65535）．

なお，カーソルキーをスペースバーといっしょに押すと，キーの操作が加速される．エンベロープ周期のように，広い値の範囲をいったりきたりするときに便利だ．

▶ A B C

操作を行うチャンネルを，A，B，C，3つのなかから選択する．選んだチャンネルの名前は，画面の上で点滅する．

▶ T E

E は現在使用中のチャンネルに対して，エンベロープのスイッチをONにする．同時に音量の表示は「16」に変わる．T は，反対にエンベロープをOFFにするもので，これによって音量はもとにもどる．

▶ N M

ノイズのON／OFF．N はそのチャンネルにノイズをのせ，M はそのチャンネルからノイズを消す．

▶ W

エンベロープの波の型を変える．W を押すと，「WAVE NUMBER（0～15）」

と表示されるから，数字をタイプして RETURN キーを押す．ここでは「0」から「15」が使えるが，0～3は9と同じ，4～7は15と同じ型になる．これは3章で説明した．

▶ R

PSGの各レジスタの状態を表示するキー．R を押すと，画面に一覧表が出る．さらに，右のほうに「Test(y，n)?」と表示されるので，Y を押すと自動演奏(!)がはじまる(もちろん，音色は現在セットされているものが使われる)．

右上にS＝8とか，M＝300とか出ているのは，PLAY文で同じエンベロープを使いたい場合に，ミュージックマクロとして設定する値だ．

N を押したときには，すぐにもとの画面にもどる．また，自動演奏を止めたいときには，S キーを押す．

▶ G

エンベロープの型によっては，一発音を出したきり黙ってしまうものもあるので，G を押すともう一度聞き直すことができる．

▶ L

通常の「ドレミ」を，SOUND文のデータに変換するキー．L を押したのち，質問にこたえてオクターブ(1～8)と音階(c～b)を指定してやると，各レジスタに，対応するデータがセットされる．和音をピタリとキメたいときに便利．

▶ D

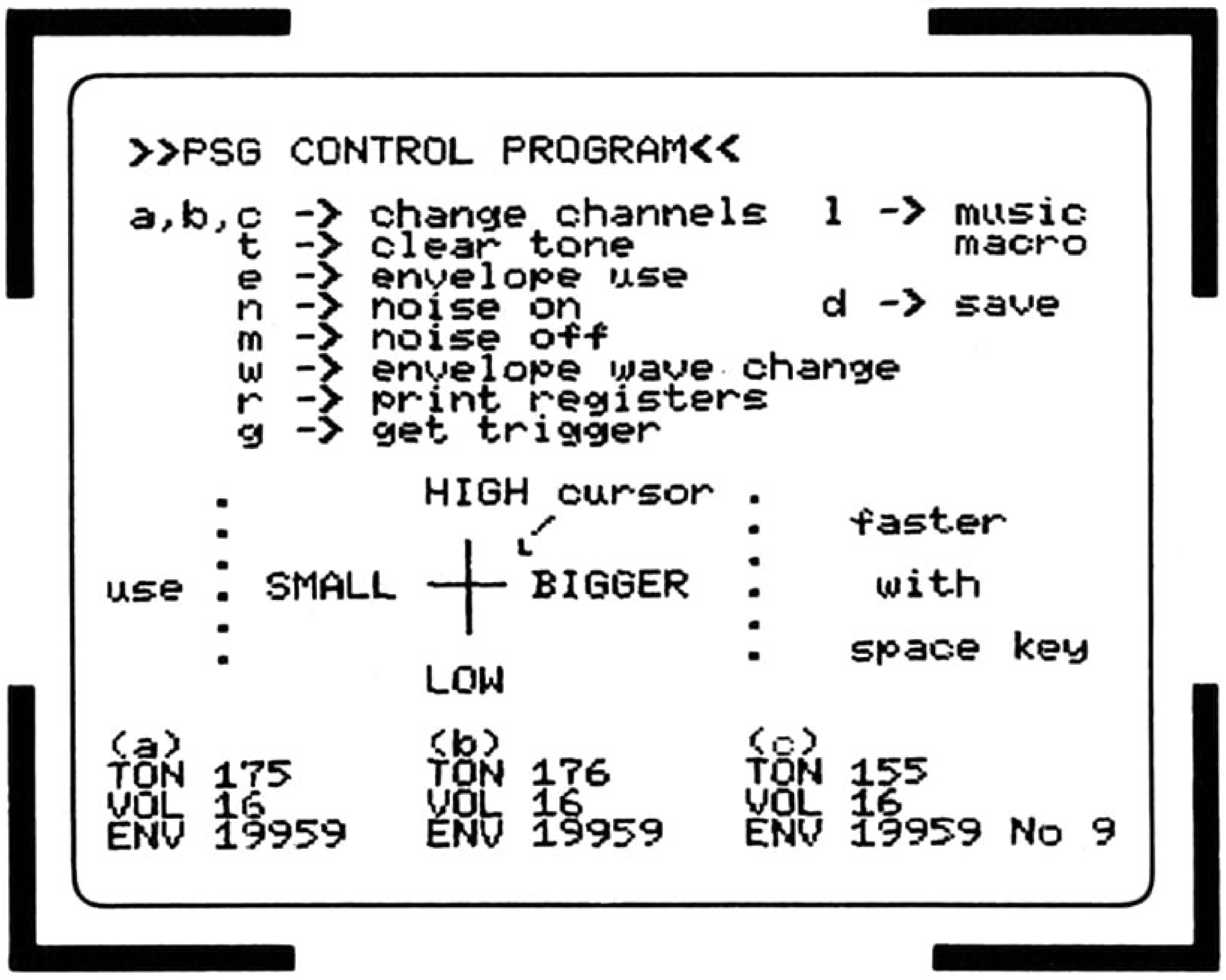
でき上がったデータをテープに保存するキー．ただ保存するだけではなく，なんとBASICのSOUND命令の集まりとして，プログラムの形でセーブする．

D を押すと，ファイル名と行番号を聞いてくるので，テープを録音状態にして，適当にこたえる．こうしてセーブしたプログラムは，あとでほかのプログラムにサブルーチンとしてマージ(合体)させることもできる．

たとえば，ゲームなどのプログラムを作っておいて，「MERGE(マージ) "CAS：ファイル名"」を実行すると，セーブのときに指定した行番号に，SOUND文がはいってくる．あとは，もとのプログラムの中で，このSOUND文をサブルーチンとし

て呼び出せばよい．

ファイル名は覚えやすい名前をつけ，行番号は，はじめは大きくしておいて，プログラムに入れてから適当に変えるとよいだろう．



では，このツールを使って，昔なつかしい「ボンボン時計」の音を作ってみよう．

まず，カーソルキーでA，B，C 3つのチャンネルの音量を適当に上げ，周波数を少しずつずらす．高さは，A＝175，B＝176，C＝155くらいがよいだろう．なにやら雅楽のような，異様な雰囲気になったはずだ．

次に，E キーで全部のチャンネルにエンベロープをかけたのち，W キーで「WAVE NUMBER(0～15)＝9」を選ぶ．さらにカーソル・キーでエンベロープの周期を20000前後に設定する．

これで，G キーを押してみよう．「コーン」と卓上の小さな時計のベルのような音が出る．どうだい，デジタルだって，PSG があれば，こんな時計の音が出せる．

この音ではどうも安っぽくていけねェ，という人は，3つのチャンネルの高さの幅をぐっと広げて低くしてみよう．同時にエンベロープの周期も長くして，重厚な感じを出す．　A＝570，B＝910，C＝1150，エンベロープ周期＝40000くらいに設定してやると，大きな柱時計の感じがでてくる．いろいろ試してみよう．

サウンド・エディター1

```
100 '----------------------------
110 '
120 '       SOUND-Editor 1
130 '
140 '----------------------------
150 '
160 '   INIT.
170 '
180 SCREEN 0
190 CLEAR &H1FF:CLS:LL=10000
200 DIM DA(13),CD(13),MD(69),NA$(4),DM$(13)
210 FOR I=0 TO 13:READ A
220 CD(I)=A:NEXT I
230 FOR I=0 TO 13:READ A$
240 DM$(I)=A$:NEXT I
250 KEY 1,"DATAGEN":KEY(1) ON
260 KEY 4,"  END   ":KEY(4) ON
270 ON KEY GOSUB 860,,,1050
280 FOR I=0 TO 13
290 SOUND I,0:NEXT I
300 '
310 '    START.
320 '
330 MM=1:GOTO 430
340 FOR K=0 TO 13
350 LOCATE 0,20:PRINT SPACE$(35)
360 LOCATE 3,20:PRINT K;
370 INPUT "DATA KEY IN PLEASE";DA(K)
380 IF DA(K)<0 OR DA(K)>CD(K) THEN LN=K+3:GOSUB 700:G
OTO 350
390 LOCATE 32,K+2:PRINT DA(K):NEXT K
400 LOCATE 0,20:PRINT SPACE$(35)
410 FOR I=0 TO 13:SOUND I,DA(I):NEXT I
420 RETURN
430 '
440 '  SCREEN SUB
450 '
460 CLS:LOCATE 0,1
470 PRINT "┌────────────┬──┬──────────────┬─────┐"
480 FOR I=0 TO 13:I$="0"+STR$(I)
490 I$=RIGHT$(I$,2)
500 PRINT "|PSG ﾚｼﾞｽﾀNO.|";I$;"|";DM$(I);"|";DA(I);
510 LOCATE 37,I+2:PRINT "|"
520 NEXT I
530 PRINT "└────────────┴──┴──────────────┴─────┘"
```

```
540 IF MM=1 THEN MM=0:GOSUB 340:GOTO 460
550 FOR I=0 TO 13:SOUND I,DA(I):NEXT I
560 IF R=1 THEN R=0:GOTO 330
570 LOCATE 0,18:PRINT "EDIT (HELP=H)";TAB(31);
580 INPUT A$
590 IF A$="H" THEN GOSUB 1240
600 LN=CSRLIN
610 IF(LN<3) OR (LN>16) THEN BEEP:GOTO550
620 A$=RIGHT$(A$,6):A$=LEFT$(A$,3)
630 DA(LN-3)=VAL(A$)
640 IF DA(LN-3)<0 OR DA(LN-3)>CD(LN-3) THEN GOSUB 700
:GOTO 550
650 FOR I=0 TO 13
660 SOUND I,DA(I):NEXT I
670 LOCATE 32,LN-1:PRINT DA(LN-3)
680 GOTO 550
690 '
700 ' ERR RECOVER
710 '
720 DA(LN-3)=0
730 LOCATE 33,LN-1:PRINT "0    ";
740 FOR I=1 TO 20
750  LOCATE 0,18
760 PRINT "DATAﾉ ｱﾀｲ ｶﾞ ﾊﾝｲｦ ｺｴﾃｲﾏｽ!!"
770 PRINT "ｱﾀｲ ﾉ ﾊﾝｲ ﾊ 0-";CD(LN-3);"ﾃﾞｽ!!"
780 BEEP:LOCATE 0,18
790 PRINT  SPACE$(35)
800 NEXT I
810 LOCATE 0,19
820 PRINT  SPACE$(35)
830 FOR I=0 TO 13:SOUND I,DA(I):NEXT I
840 RETURN
850 '
860 ' DATA GEN.
870 '
880 X1=POS(0):Y1=CSRLIN
890 LOCATE 0,19
900 INPUT "DATA NAME ";NA$(H)
910 LOCATE 0,19
920 PRINT SPACE$(36):LOCATE X1,Y1
930 H=H+1
940 FOR I=0 TO 13
950 MD(J)=DA(I)
960 J=J+1
970 NEXT I
980 R=1
990 IF J<69 THEN FOR I=0 TO 13:DA(I)=0:NEXT I:RETURN
```

```
1000 FOR I=1 TO 20
1010 LOCATE 0,20:PRINT "DATA FULL!!"
1020 BEEP:LOCATE 0,20
1030 PRINT SPACE$(15):NEXT I
1040 '
1050 'END SUB
1060 '
1070 CLS
1080 FOR J=0 TO 4
1090 A$=STR$(LL)+" '--- "+NA$(J)
1100 PRINT A$
1110 LL=LL+10
1120 A$=STR$(LL)+" DATA "
1130 FOR I=0 TO 13
1140 A$=A$+STR$(MD(K))+","
1150 K=K+1
1160 NEXT I
1170 PRINT A$;CHR$(29);" "
1180 LL=LL+10
1190 NEXT J
1200 KEY 1,"color":KEY 4,"list"+CHR$(13)
1210 KEY 4,"list"+CHR$(13)
1220 END
1230 '
1240 '    HELP .
1250 '
1260 LOCATE 0,17
1270 PRINT "* ｶｰｿﾙ ｦ EDIT ｼﾀｲ DATA ﾉ ﾄｺﾛ ﾍ"
1280 PRINT "ｳｺﾞｶｼ EDIT ｼﾃ(REURN)ｷｰ ｦ ｵｾ｡"
1290 PRINT "* DATA ﾉ ｻｸｾｲ ﾊ(F1)ｷｰ ｦ"
1300 PRINT "ｵﾜﾙ ﾄｷ ﾆﾊ (F5)ｷｰ ｦ"
1310 PRINT "ｿﾚｿﾞﾚ ｵｼﾃｶﾗ(RETURN)ｷｰｦ ｵｾ｡"
1320 FOR I=1 TO 2000:NEXT I
1330 FOR I=17 TO 21
1340 LOCATE 0,I:PRINT SPACE$(35)
1350 NEXT I:A$=""
1360 LOCATE 0,18:RETURN
1370 DATA 255,15,255,15,255,15,31
1380 DATA 63,16,16,16,255,255,15
1390 DATA CH-A FINE TUNE,CH-A  --- TUNE
1400 DATA CH-B FINE TUNE,CH-B  --- TUNE
1410 DATA CH-C FINE TUNE,CH-C  --- TUNE
1420 DATA NoiseFrequency,CH       SELECT
1430 DATA CH-A     VOLUME,CH-B    VOLUME
1440 DATA CH-C     VOLUME,ENV. FINE TUNE
1450 DATA ENV.        TUNE,ENV.   PATTERN
```

サウンド・エディター2

```
100 ' ------------------------
110 '
120 '     SOUND-EDITOR 2
130 '
140 ' ------------------------
150 CLEAR 300:TGL=1:GOSUB 220:GOSUB 330
160 FOR I=0 TO 13:SOUND I,REG(I):NEXT
170 FOR I=0 TO 2:PUSH(I)=0:NEXT
180 GOTO 730
190 '
200 ' initialize
210 '
220 DIM REG(13),X(3,6),L(3,6),M(3,6),T(7),T$(7),PUSH(2)
230 RESTORE 260:FOR I=0 TO 13:READ DA: REG(I)=DA:NEXT
240 RESTORE 270:FOR I=1 TO 7:READ DA:T(I)=DA:NEXT
250 RESTORE 280:FOR I=1 TO 7:READ DA$: T$(I)=DA$:NEXT
260 DATA 90,0,90,0,90,0,15,56,0,0,0,44,1,8
270 DATA .55,.6174,.327,.3671,.412,.4365,.49
280 DATA c,d,e,f,g,a,b
290 RETURN
300 '
310 ' screen
320 '
330 KEY OFF:SCREEN 0,,0:WIDTH 39:COLOR 15,4:CLS
340 GOSUB 380:GOSUB 520:GOSUB 610:RETURN
350 '
360 ' scr1
370 '
380 PRINT " >>PSG CONTROL PROGRAM<<"
390 PRINT
400 PRINT " a,b,c -> change channels  l -> music"
410 PRINT "     t -> clear tone            macro"
420 PRINT "     e -> envelope use"
430 PRINT "     n -> noise on           d -> save"
440 PRINT "     m -> noise off"
450 PRINT "     w -> envelope wave change"
460 PRINT "     r -> print registers"
470 PRINT "     g -> get trigger"
480 RETURN
490 '
500 ' scr2
510 '
```

```
520 PRINT
530 PRINT "    .      HIGH cursor ."
540 PRINT "    .           /      .   faster"
550 PRINT "    .         | L      ."
560 PRINT "use . SMALL -+- BIGGER .    with"
570 PRINT "    .         |        ."
580 PRINT "    .                  .   space key"
590 PRINT "           LOW"
600 PRINT:RETURN
610 PRINT "(a)        (b)         (c)"
620 PRINT "TON 90     TON 90      TON 90"
630 PRINT "VOL 0      VOL 0       VOL 0"
640 '
650 ' mode indicate
660 '
670 TGL=-TGL
680 IF TGL=1 THEN LOCATE 12*CN+1,19:PRINT " ";:RETURN
690 IF TGL=-1 THEN LOCATE 12*CN+1,19:PRINT CHR$(CN+97)
:RETURN
700 '
710 ' key sens.routine
720 '
730 CAT$=INKEY$:CUR=STICK(0):BAR=-10*(STRIG(0)=-1)
740 '
750 ' high
760 '
770 IF CUR<>1 THEN 860
780 R=CN*2:X=REG(R)+REG(R+1)*256-1-BAR
790 IF X<0 THEN X=0
800 REG(R)=X MOD 256:REG(R+1)=X ¥ 256
810 SOUND R,REG(R):SOUND R+1,REG(R+1)
820 LOCATE 3+CN*12,20:PRINT X:GOTO 730
830 '
840 ' low
850 '
860 IF CUR<>5 THEN 940
870 R=CN*2:X=REG(R)+REG(R+1)*256+1+BAR
880 IF X>4095 THEN X=4095
890 REG(R)=X MOD 256:REG(R+1)=X ¥ 256
900 GOTO 810
910 '
920 ' bigger
930 '
940 IF CUR<>3 THEN 1160
950 X=REG(CN+8):IF X=16 THEN 1060
960 '
```

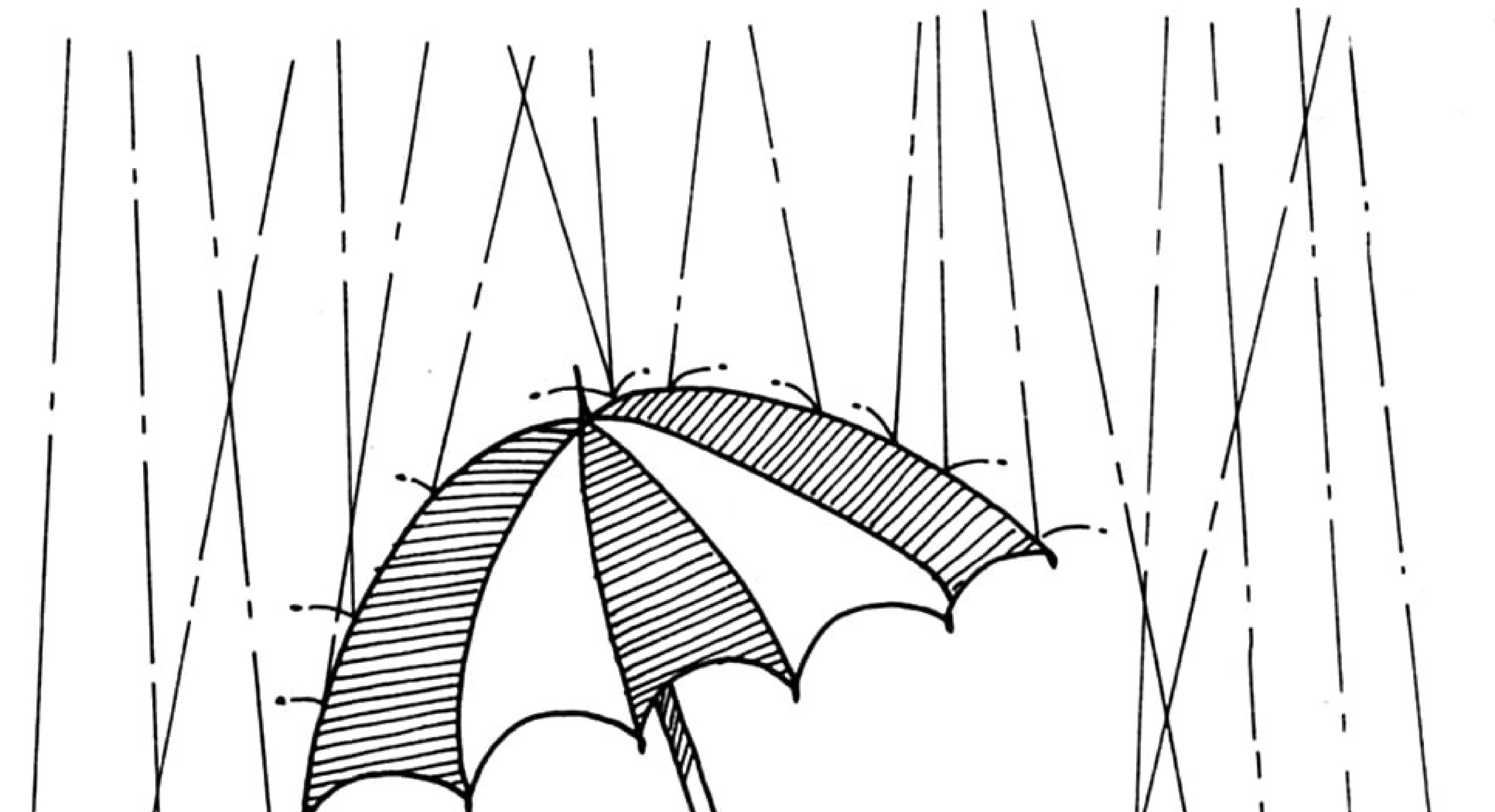
```
970 ' big.vol
980 '
990 X=X+1+BAR/10:IF X>15 THEN X=15
1000 REG(CN+8)=X:PUSH(CN)=X
1010 LOCATE 3+CN*12,21:PRINT X
1020 SOUND 8+CN,REG(8+CN):GOTO 730
1030 '
1040 ' big.env
1050 '
1060 X=REG(11)+REG(12)*256+1+BAR*9
1070 IF X>65535! THEN X=65535!
1080 REG(12)=INT(X/256):REG(11)=X-REG(12)*256
1090 FOR I=0 TO 2
1100 IF REG(I+8)=16 THEN LOCATE 3+I*12,22:PRINT X
1110 NEXT
1120 SOUND 11,REG(11):SOUND 12,REG(12):GOTO 730
1130 '
1140 ' smaller
1150 '
1160 IF CUR<>7 THEN 1370
1170 X=REG(CN+8):IF X=16 THEN 1280
1180 '
1190 ' small.vol
1200 '
1210 X=X-1-BAR/10:IF X<0 THEN X=0
1220 REG(CN+8)=X:PUSH(CN)=X
1230 LOCATE 3+CN*12,21:PRINT X
1240 SOUND 8+CN,REG(8+CN):GOTO 730
1250 '
1260 ' small.env
1270 '
1280 X=REG(11)+REG(12)*256-1-BAR*9:IF X<0 THEN X=0
1290 REG(12)=INT(X/256):REG(11)=X-REG(12)*256
1300 FOR I=0 TO 2
1310 IF REG(I+8)=16 THEN LOCATE 3+I*12,22:PRINT X
1320 NEXT
1330 SOUND 11,REG(11):SOUND 12,REG(12):GOTO 730
1340 '
1350 ' channel
1360 '
1370 IF CAT$>"c" THEN 1490
1380 IF CAT$<"a" THEN GOSUB 650:GOTO 730
1390 CN=ASC(CAT$)-97
1400 GOSUB 650
1410 LOCATE 0,19
1420 PRINT "(a)"SPC(9)"(b)"SPC(9)"(c)"
1430 GOTO 730
```

```
1440 '
1450 '  character-sens.
1460 '
1470 '  't'
1480 '
1490 IF CAT$<>'t' THEN 1590
1500 REG(CN+8)=PUSH(CN):GOSUB 2670
1510 LOCATE CN*12,22:PRINT SPC(10)
1520 LOCATE CN*12,20:PRINT 'TON';REG(CN*2)+REG(CN*2+1)*256
1530 LOCATE 3+CN*12,21:PRINT REG(CN+8)
1540 LOCATE CN*12,22:PRINT SPC(10);
1550 GOTO 730
1560 '
1570 '  'e'
1580 '
1590 IF CAT$<>'e' THEN 1670
1600 PUSH(CN)=REG(CN+8):REG(CN+8)=16:GOSUB 2670
1610 LOCATE 3+CN*12,21:PRINT ' 16'
1620 LOCATE CN*12,22:PRINT 'ENV';REG(11)+REG(12)*256
1630 GOTO 730
1640 '
1650 '  'n'
1660 '
1670 IF CAT$<>'n' THEN 1740
1680 REG(7)=REG(7)AND(NOT8*(2^CN)):GOSUB 2670
1690 LOCATE CN*12,23:PRINT 'NOISE';
1700 GOTO 730
1710 '
1720 '  'm'
1730 '
1740 IF CAT$<>'m' THEN 1810
1750 REG(7)=REG(7)OR(8*(2^CN)):GOSUB 2670
1760 LOCATE CN*12,23:PRINT SPC(5);
1770 GOTO 730
1780 '
1790 '  'g'
1800 '
1810 IF CAT$='g' THEN GOSUB 2670:GOTO 730
1820 '
1830 '  'w'
1840 '
1850 IF CAT$<>'w' THEN 2040
1860 LOCATE 0,18
1870 PRINT 'WAVE NUMBER(0~15)';:X$=INKEY$
1880 IF X$='' OR X$<'0' OR X$>'9' THEN 1860
1890 PRINT '=';X$;
```

```
1900 Y$=INKEY$:IF Y$=CHR$(13) THEN 1950
1910 IF Y$<'0' OR Y$>'9' OR Y$='' THEN 1900
1920 IF VAL(X$+Y$)>15 THEN 1900
1930 PRINT Y$;:X$=X$+Y$
1940 IF INKEY$<>CHR$(13) THEN 1940
1950 REG(13)=VAL(X$):GOSUB 2670
1960 X=0:FOR I=8 TO 10
1970 IF REG(I)=16 THEN X=1
1980 NEXT:LOCATE 0,18
1990 PRINT SPC(20);:IF X=0 THEN 730
2000 LOCATE 34,22:PRINT'No';VAL(X$);:GOTO 730
2010 '
2020 ' 'r'
2030 '
2040 IF CAT$<>'r' THEN 2490
2050 FOR I=2 TO 11:LOCATE 0,I:PRINT SPC(38);:NEXT I
2060 LOCATE 7,2:PRINT 'REGISTERS':PRINT
2070 FOR I=0 TO 9:PRINT 'reg';STR$(I);' =';STR$(REG(I)),
: NEXT
2080 FOR I=10 TO 13:PRINT 'reg';STR$(I);'=';STR$(REG(I))
,:NEXT
2090 SS$='S ='+STR$(REG(13)):MM$='M ='+STR$(REG(11)+REG(
12)*256)
2100 IF REG(11)+REG(12)=0 THEN MM$='M = 1'
2110 LOCATE 28,4:PRINT SS$;
2120 LOCATE 28,5:PRINT MM$;
2130 LOCATE 28,7:PRINT 'Test(y,n)?';
2140 X$=INKEY$:IF X$='n' THEN 2420
2150 IF X$<>'y' THEN 2140
2160 LOCATE 28,9:PRINT 's -> stop'
2170 FOR I=0 TO 2
2180 SS$(I)='V15T':MM$(I)=''
2190 IF REG(I+8)=16 THEN SS$(I)='S'+STR$(REG(13))
2200 IF REG(I+8)=16 THEN MM$(I)='M'+STR$(REG(11)+REG(12)
*256)
2210 IF RIGHT$(MM$(I),1)='0' THEN MM$(I)='M1'
2220 SS$(I)=SS$(I)+MM$(I)
2230 NEXT
2240 PLAY SS$(0),SS$(1),SS$(2)
2250 X=RND(-TIME):FOR H=1 TO 3:FOR I=1 TO 6:X(H,I)=1:NEX
T:NEXT
2260 '
2270 ' auto-play
2280 '
2290 FOR H=1 TO 3:FOR I=1 TO 6
2300 X(H,I)=X(H,I)-1
```

```
2310 IF X(H,I)=0 THEN X(H,I)=2^(I-1):L(H,I)=INT(RND(1)
*2):M(H,I)=INT(RND(1)*11)
2320 NEXT:NEXT
2330 FOR H=1 TO 3
2340 LL(H)=2^(L(H,2)+L(H,3)+L(H,4)+1)
2350 MM(H)=20+M(H,1)+M(H,2)+M(H,3)+M(H,4)+M(H,5)+M(H,6)
2360 NEXT
2370 PLAY "l=ll(1);n=mm(1);","l=ll(2);n=mm(2);","l=ll(3
);n=mm(3);"
2380 IF INKEY$<>"s" THEN 2290
2390 '
2400 ' clear
2410 '
2420 LOCATE 27,9:PRINT "please wait"
2430 IF PLAY(0) THEN 2430
2440 FOR I=2 TO 11:LOCATE 0,I:PRINT SPC(38);:NEXT I
2450 LOCATE 0,2:GOSUB 400:GOSUB 520:GOSUB 2670:GOTO 730
2460 '
2470 ' "l"
2480 '
2490 IF CAT$<>"l" THEN 2790
2500 LOCATE 0,11:PRINT SPC(200);SPC(73):LOCATE 0,11
2510 FOR I=0 TO 2
2520 PRINT "Ch.";CHR$(97+I);" Oct.(1~8)=";
2530 X$=INKEY$:IF X$<"1" OR X$>"8" THEN 2530
2540 PRINT X$:X=VAL(X$)
2550 PRINT "      Music Macro (cdefgab)=";
2560 Y$=INKEY$:IF Y$<"a" OR Y$>"g" THEN 2560
2570 PRINT Y$:Y=T(ASC(Y$)-96)
2580 Z=19967.5/(16*Y*(2^X))
2590 REG(I*2+1)=INT(Z/256):REG(I*2)=INT(Z-REG(I*2+1)*256)
2600 NEXT I
2610 FOR I=0 TO 2:LOCATE I*12+3,20:PRINT REG(I*2)+REG(I*2
+1)*256:NEXT
2620 GOSUB 2670:LOCATE 0,11:PRINT SPC(200);SPC(73)
2630 LOCATE 0,10:GOSUB 520:GOTO 730
2640 '
2650 ' reg-set
2660 '
2670 FOR I=0 TO 13
2680 IF I<>7 THEN SOUND I,REG(I)
2690 NEXT
2700 SOUND 7,REG(7)
2710 FOR I=8 TO 10
2720 IF REG(I)=16 THEN LOCATE 34,22:PRINT "No";REG(13);:R
ETURN
```

```
2730 NEXT
2740 LOCATE 34,22:PRINT SPC(5);
2750 RETURN
2760 '
2770 ' 'd'
2780 '
2790 IF CAT$<>'d' THEN GOSUB 650:GOTO 730
2800 FOR I=2 TO 9:LOCATE 0,I:PRINT SPC(38);:NEXT I
2810 LOCATE 2,3:PRINT '-- BASIC ﾌﾟﾛｸﾞﾗﾑ ｾｰﾌﾞ --';
2820 LOCATE 5,5:PRINT 'ｶｾｯﾄ ﾆ ﾛｸｵﾝ ｼﾏｽ';
2830 F$='':LN$=''
2840 LOCATE 5,7:INPUT 'FILE NAME=';F$
2850 IF F$='' THEN 2970
2860 LOCATE 5,9:INPUT 'LINE NUMBER=';LN$
2870 IF LN$='' THEN 2970
2880 LN=VAL(LN$)
2890 OPEN F$ FOR OUTPUT AS #1
2900 FOR I=0 TO 12 STEP 2
2910 A$=STR$(LN)+' SOUND '+STR$(I)+','+STR$(REG(I))
2920 A$=A$+':'+'SOUND '+STR$(I+1)+','+STR$(REG(I+1))
2930 PRINT #1,A$:LN=LN+10
2940 NEXT I
2950 CLOSE
2960 FOR I=2 TO 9:LOCATE 0,I:PRINT SPC(38);:NEXT I
2970 LOCATE 0,2:GOSUB 400:GOTO 730
```



BLOCK 5 FIVE MSXサウンド活用プログラム

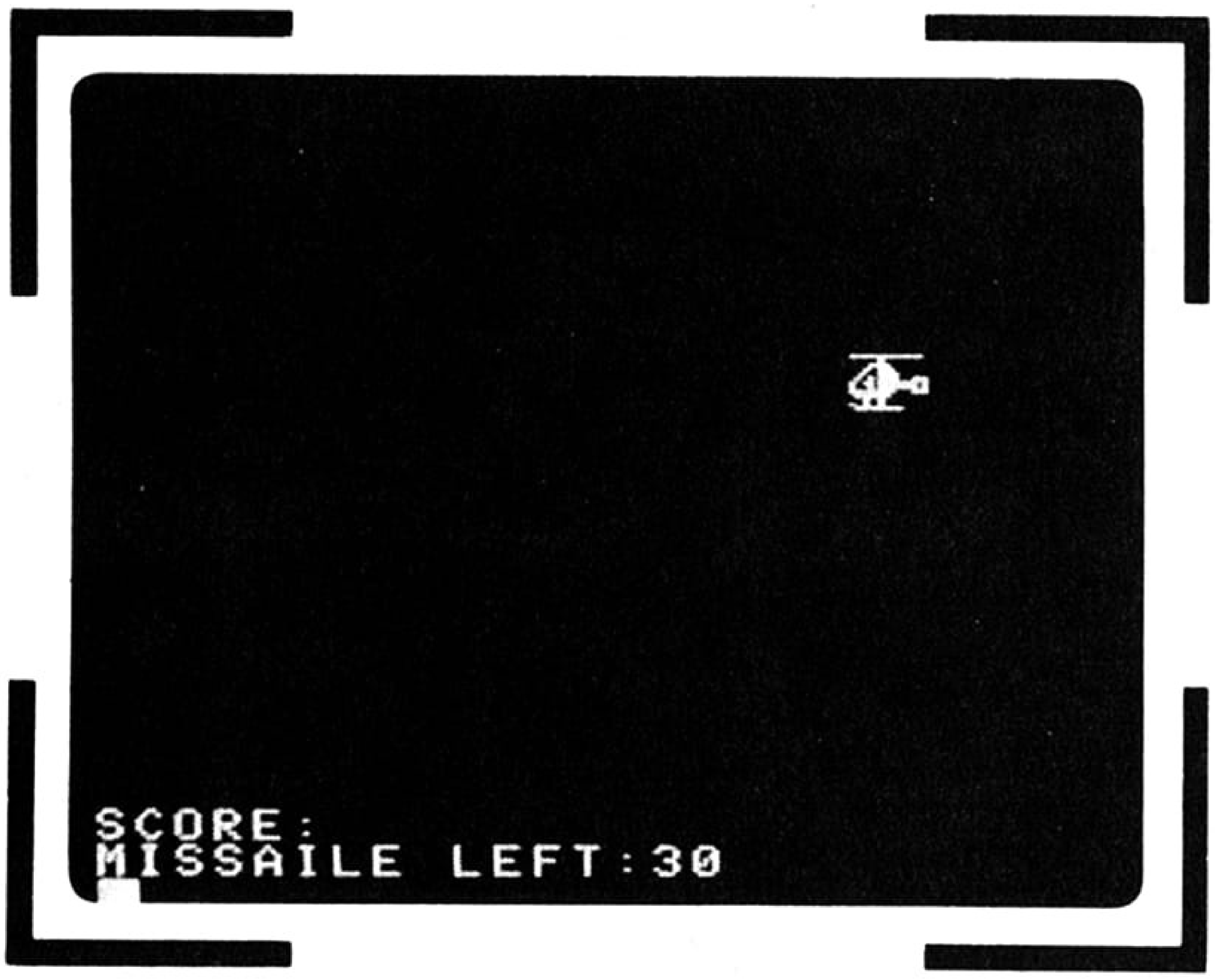
さて，こうして作った効果音を実際にプログラムの中で利用してみよう．この章では，3つのプログラム例を紹介する．画面と音をいかに組み合わせるか，そのあたりの参考にしてほしい．

5-1 ゲームに効果音を組みこむ

まずは，典型的なゲームと効果音の組み合わせをサンプルとしてお目にかけよう．

56ページのリストは，名付けて「チャップイ・リフター」．

RUN させると，ヘリコプターが右から左へと飛んでくるので，スペースバーで



ミサイルを発射して打ち落とす．ミサイルは誘導式となっていて，カーソルキーで左右にコントロールできる．

うまくヘリに命中させると，人が落っこちてきて得点アップだ．

ヘリやミサイルなどは，すべてMSXのスプライト機能を利用して表示している．スプライトについてのくわしい説明は省くが，あらかじめ決めておいたパターン(図形)を自由に画面に表示させる機能のことだ．リストの1020行から1210行まで続いているデータが，ヘリなどの形を定義するためのものだ．

このゲームでは,「ヘリの飛行音」「ミサイルの発射音」「ミサイルの命中音」「人が落下する音」「人が地面に落ちた音」の５つの効果音が使われている．５つの音のデータは，プログラムのいちばんおしまいに，まとめてならべてある．なお，「ヘリ」の音は，さきほど「エディター１」で作ったのと同じものだ．

音出しルーチンそのものは，1240行－1250行などのように，まず必要とする音のデータ行にRESTORE(リストア)をかけて，FOR(フォア)～NEXT(ネクスト)で14コいっぺんにPSGのレジスタに書きこんでいる．

```
570 RESTORE1270:FOR I=0 TO 13
580 READ D:SOUND I,D:NEXT I
```

この中で，少し変わった手を使っているのが，1670行－1720行だ．ここでは，チャンネルAのデータを「10」ずつ増やしていって(だんだん音を低くしていって),「ピュー」とものが落ちてくる感じを出している．

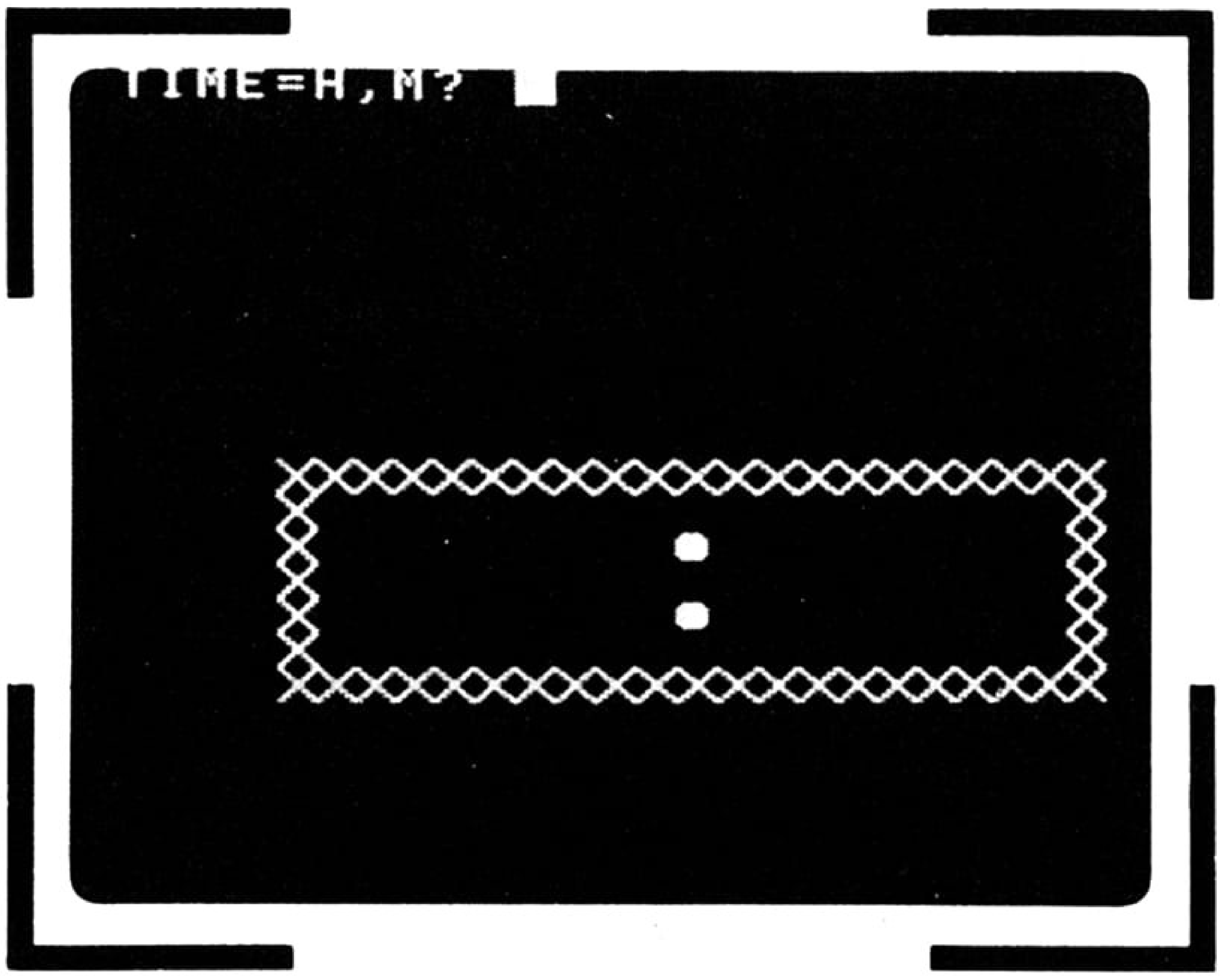
```
750 FOR I=Y TO 200
760 SOUND 0,I+10
770 PUT SPRITE 1,(X,I),15,2
780 NEXT I
```

ゲームのほうは，ミサイルを30発のうち，いくつ当てられるかが勝負だ．がんばろう！

5-2 MSXボンボン時計

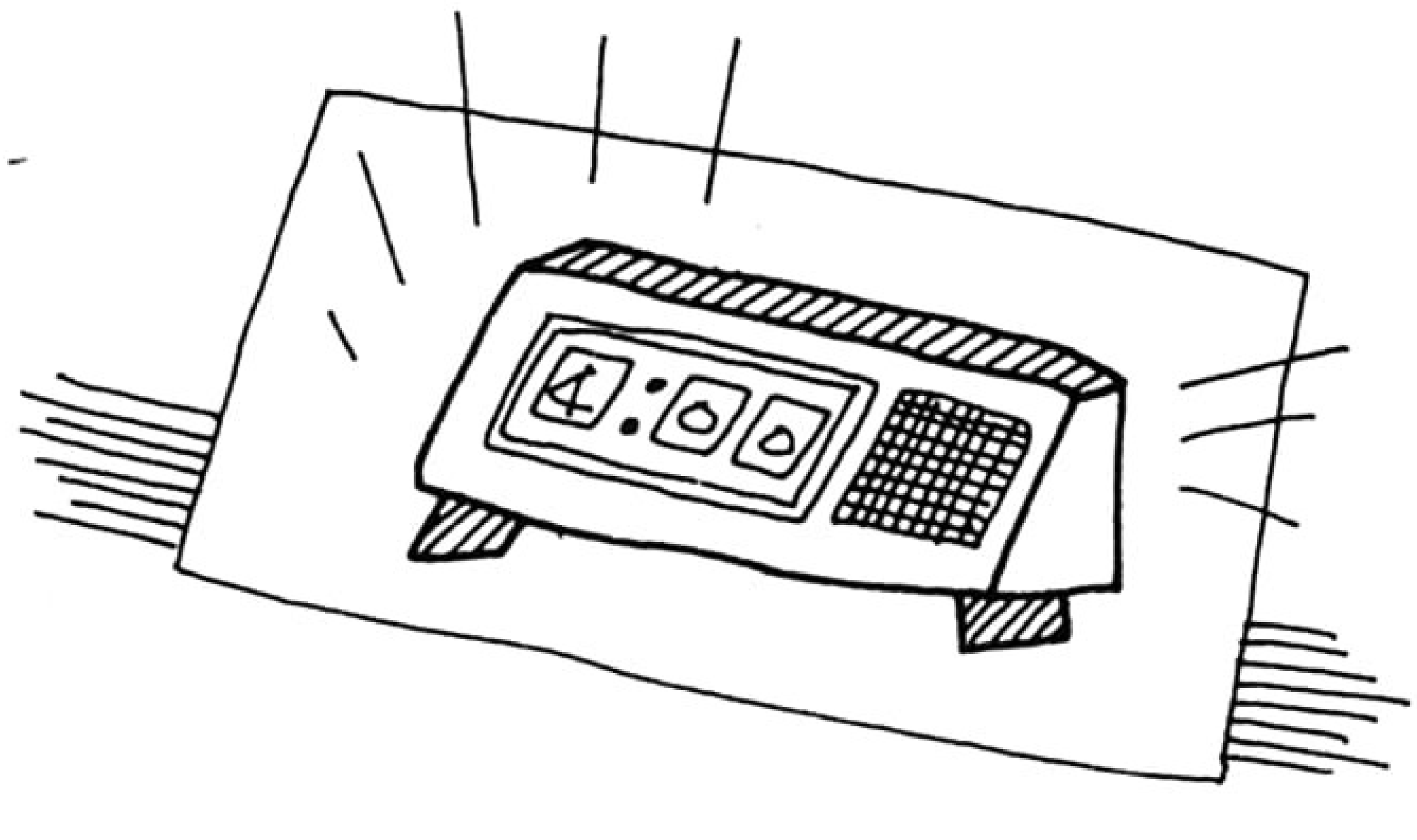
次は,「エディター2」で作った「時計」の音を利用して,時計そのものを作ってみた.

59ページのリストをRUNさせると,まず「TIME=(H, M)?」と質問がでるので,「時間, 分」とコンマで区切って入力してやる.これで,タイムセットが完了して,画面にデジタル時計が現われる.



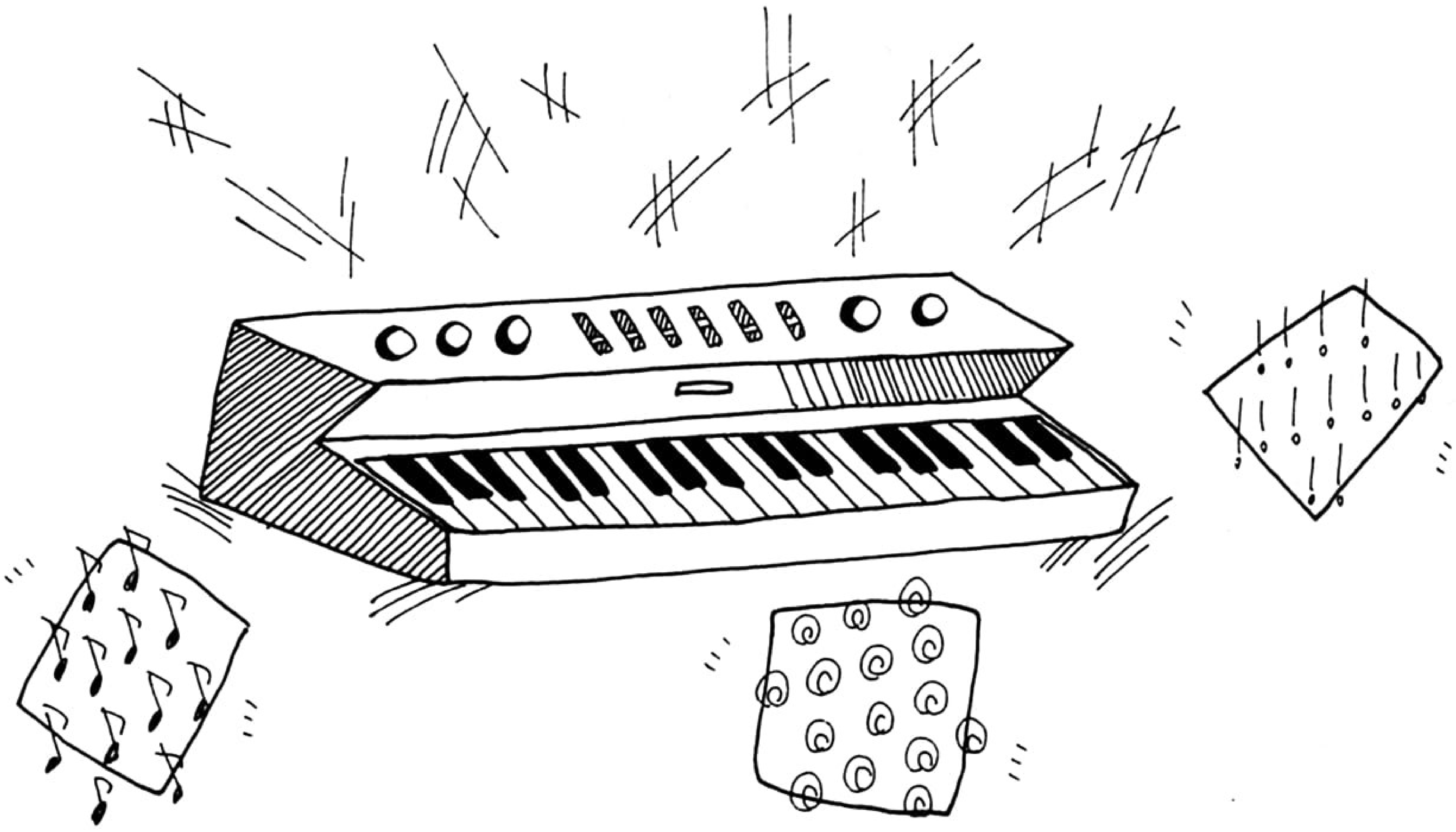
時計を「11時59分」ぐらいにセットしてみよう.12時をむかえるとともに,「ボーン,ボーン」とベルの音が鳴りだす.これを目覚まし時計のかわりに使ってもよい.もっともこんな音じゃあとっても起きられないという人は,サウンド・データを変えて,もっとけたたましい音にしよう.

なお,この時計の数字も,スプライトで表示している.





MSXリズムボックス 5-3



さて，3番目は，「MSX リズムボックス」だ(61ページリスト)．このプログラムは，絵はなにも出ないが，音出しのタイミングをとる実例としてあげておいた．

RUN すると，「トトタト，トトタト」とエイト・ビートのリズムが流れ出す．F 1 と F 2 のキーで，テンポを変えることができる．

また，パーカッションとしても使えるように，AUTO リズムのほかに，マニュアル・モードを用意しておいた．F 3 のキーを押すと，以後は，左右のカーソルキーで，2つの音をたたき分けることができる．AUTO にもどるには，上むきのカーソルキー ↑ を押せばよい．

リズムの早さの調節は，280行，310行などのFOR文の中にある「L」という変数の値を，ファンクションキーの割り込みを利用して増やしたり減らしたりすることで行なっている．「L」の後ろについている「+42」とか「−14」とかいう値はリズムの微調整だ．プログラムを走らせてみて，4つの音の間隔が詰まったり開いたりしているような気がしたら，この数値を加減して，ちゃんとした8ビートに聞こえるよう調整しよう．

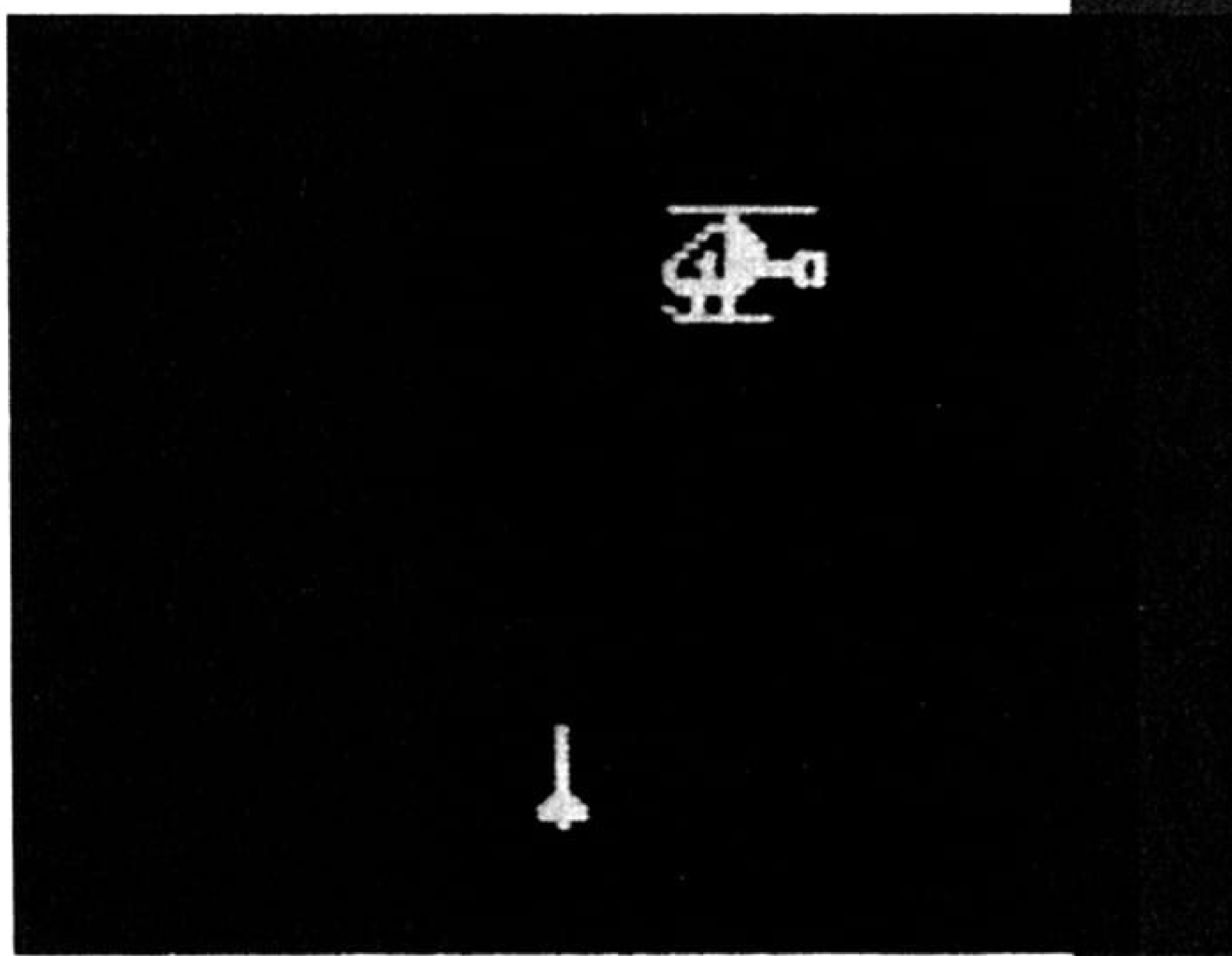
このリズムボックスは，現在のところ，1つしかリズム・パターンを持っていないが，改良しだいでは，もっとレパートリーをひろげられるだろう．

また，サウンド・データのほうを変えてやる(たとえば，音源をノイズからトーンにするなど)と，もっとひょうきんなリズムボックスになる．

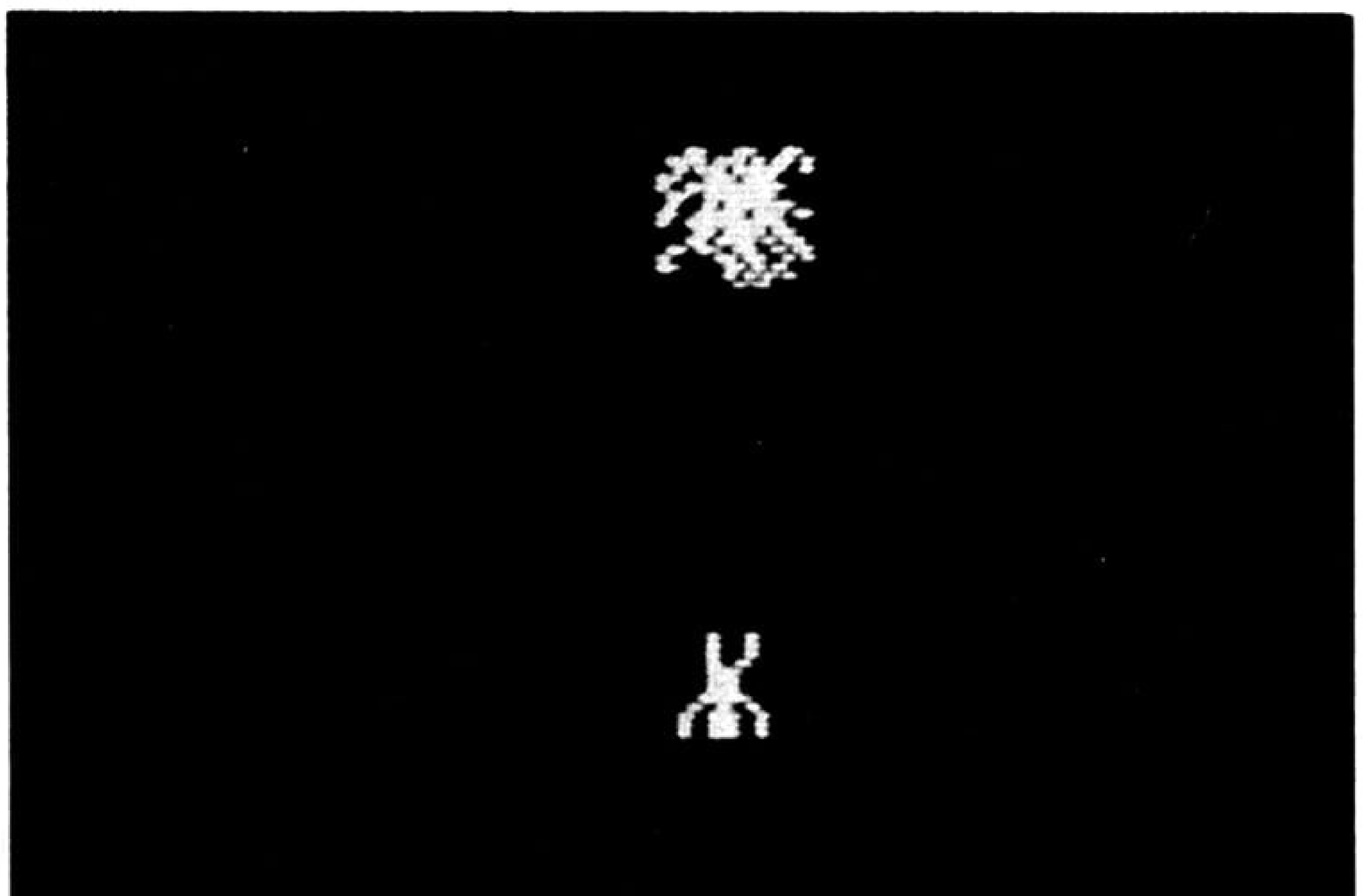
サンプルプログラム-1「チャップイ・リフター」

```
100 '-------------------
110 '
120 '  CAPPUI LIFTER
130 '
140 '-------------------
150 '
160 SCREEN 1,2:KEY OFF
170 LOCATE 0,21:PRINT "SCORE:"
180 LOCATE 0,22:PRINT "MISSILE LEFT:30"
190 FOR J=0 TO 3
200 FOR I=1 TO 32
210 READ A$:B$=B$+CHR$(VAL("&H"+A$))
220 NEXT I:SPRITE$(J)=B$
230 B$="":NEXT J
240 SPRITE ON
250 STRIG(0) ON
260 ON SPRITE GOSUB 640
270 ON STRIG GOSUB 510
280 X=200:Y=50:X1=150:Y1=200
290 MI=30
300 RESTORE 1230:FOR I=0 TO 13
310 READ D:SOUND I,D:NEXT I
320 '
330 ' MOVE HERI
340 '
350 IF X=0 THEN Y=INT(RND(.5)*100):X=250
360 PUTSPRITE 0,(X,Y),11,0
370 X=X-1
380 IF MI< 0 THEN GOTO 930
390 IF H =0 THEN GOTO 330
400 PUTSPRITE 1,(X1,Y1),13,3
410 IF STICK(0)=7 THEN X1=X1-1
420 IF STICK(0)=3 THEN X1=X1+1
430 Y1=Y1-1
440 IF Y1>0 THEN GOTO 330
450 PUTSPRITE 1,(X1,209),13,3:STRIG(0) ON
460 IF MI =0 THEN GOTO 930
470 X1=150:Y1=200:H=0
480 SPRITE ON
490 GOTO 330
500 '
```

```
510 ' MISSILE ON.
520 '
530 IF MI=0 THEN STRIG(0) OFF
540 H=1:MI=MI-1
550 LOCATE0,22:PRINT "MISSILE LEFT:";MI;"  "
560 SPRITE ON:STRIG(0) OFF
570 RESTORE1270:FOR I=0 TO 13
580 READ D:SOUND I,D:NEXT I
590 FOR I=1 TO 200:NEXT I
600 RESTORE 1230:FOR I=0 TO 13
610 READ D:SOUND I,D:NEXT I
620 RETURN
630 '
640 ' BAKUHATU SUB.
650 '
660 PUT SPRITE 0,(X,209),11,0
670 PUT SPRITE 1,(X,209),13,3
680 PUT SPRITE 0,(X,Y),6,1
690 RESTORE 1250:FOR I=0 TO 13
700 READ D:SOUND I,D:NEXT I
710 SPRITE OFF
720 FOR I=1 TO 400:NEXT I
730 RESTORE 1290
740 FOR I=0 TO 13:READ D:SOUND I,D:NEXT I
750 FOR I=Y TO 200
760 SOUND 0,I+10
770 PUT SPRITE 1,(X,I),15,2
780 NEXT I
790 RESTORE 1310
800 FOR I=0 TO 13:READ D:SOUND I,D:NEXT I
810 FOR I=1 TO 100:NEXT I
820 PUT SPRITE 0,(X,209),6,1
830 PUT SPRITE 1,(X,209),15,2
840 H=0:X1=150:Y1=200
850 SC=SC+(100- INT(Y/10 )*10)
860 LOCATE 5,21:PRINT SC
870 RESTORE 1230:FOR I=0 TO 13
880 READ D:SOUND I,D:NEXT I
890 STRIG(0) ON
900 IF MI=0 THEN 950
910 RETURN
920 '
930 ' END SUB.
940 '
950 KEY 5,"RUN"+CHR$(13)
960 FOR I=0 TO 13:SOUND I,0
```



```
970 NEXT I:LOCATE 8,10
980 PRINT "GAME OVER!!"
990 LOCATE 5,12:PRINT "<AGAIN?(YES-(F5))"
1000 PRINT CHR$(11)
1010 END
1020 ' jet heri
1030 DATA 00,00,00,FF,02,0F,13,23
1040 DATA 4B,9B,8B,FF,7F,12,92,7F
1050 DATA 00,00,00,FE,00,00,80,C0
1060 DATA C7,FD,FD,87,00,00,00,E0
1070 'bakuhatu
1080 DATA 18,7B,2E,C7,9F,67,45,CF
1090 DATA 9F,0D,1B,6C,83,C1,01,00
1100 DATA C6,AD,F9,70,F8,F0,FC,BB
1110 DATA F8,EC,D6,CB,65,A8,54,A0
1120 ' HITO
1130 DATA 00,04,04,04,04,07,07,07
1140 DATA 0F,12,27,27,27,00,00,00
1150 DATA 00,40,40,40,80,00,00,00
1160 DATA 80,40,20,20,20,00,00,00
1170 ' MISSILE
1180 DATA 04,04,04,04,04,04,04,04
1190 DATA 0E,1F,1F,04,00,00,00,00
1200 DATA 00,00,00,00,00,00,00,00
1210 DATA 00,00,00,00,00,00,00,00
1220 'JET HERI.
1230 DATA 0,0,24,0,0,0,27,69,16,16,0,133,1,12
1240 'BOMB
1250 DATA 111,2,24,0,0,0,20,55,16,0,0,239,60,9
1260 'MISSILE HATUSHIYA
1270 DATA 155,0,124,0,49,0,23,46,16,16,16,59,15,15
1280 'TENRAKU
1290 DATA 55,0,0,0,0,0,0,62,10,0,0,0,0,0
1300 'GEKITOTU
1310 DATA 255,150,0,0,0,0,8,54,16,0,0,10,11,9
```



サンプルプログラム-2「MSXボンボン時計」

```
100 '-----------------
110 '
120 '      CLOCK
130 '
140 '-----------------
150 '
160 SCREEN 1,3:CLEAR &H300
170 KEY OFF
180 FOR I=0 TO 13
190 READ A:SOUND I,A
200 NEXT I
210 FOR I=0 TO 9
220 FOR J=1 TO 32
230 READ A$
240 B$=B$+CHR$(VAL(A$))
250 NEXT J
260 SPRITE$(I)=B$
270 B$="":NEXT I
280 LOCATE 4,11:PRINT "XXXXXXXXXXXXXXXXXXXXX"
290 LOCATE 4,12:PRINT "X                   X"
300 LOCATE 4,13:PRINT "X         ●         X"
310 LOCATE 4,14:PRINT "X                   X"
320 LOCATE 4,15:PRINT "X         ●         X"
330 LOCATE 4,16:PRINT "X                   X"
340 LOCATE 4,17:PRINT "XXXXXXXXXXXXXXXXXXXXX"
350 '
360 ' MAIN
370 '
380 PRINT CHR$(11);
390 INPUT "TIME=H,M";H,M
400 IF (H>12) OR (H<1) OR (M>59) OR (M<1) THEN 380
410 PRINT CHR$(11);SPACE$(25)
420 IF H=0 THEN H=10:M=10
430 TIME=0
440 T=TIME
450 S=INT(T/360)
460 IF S=10 THEN TIME=0:M=M+1
470 IF M=60 THEN M=0:H=H+1:F=1
480 IF H=13 THEN H=1
490 IF M<10 THEN M1=0:M2=M:GOTO 510
500 M2=VAL(RIGHT$(STR$(M),1)):M1=(M-M2)/10
510 IF H<10 THEN H1=0:H2=H:GOTO 530
520 H2=VAL(RIGHT$(STR$(H),1)):H1=(H-H2)/10
530 PUT SPRITE 0,(170,100),8,M2
```

```
540 PUT SPRITE 1,(140,100),8,M1
550 PUT SPRITE 2,(90 ,100),8,H2
560 PUT SPRITE 3,(60 ,100),8,H1
570 IF F=1 THEN GOSUB 590
580 GOTO 440
590 ' ｱﾗｰﾑ
600 FOR I=1 TO H
610 SOUND 13,9
620 FOR J=1 TO 600 :NEXT J
630 NEXT I:F=0:RETURN
640 DATA 193,0,255,0,226,0,0,56,16,16,16,103,94,0
650 DATA 31,32,64,64,64,64,64,64,64,64,64,64,64,64,32,31
660 DATA 248,4,2,2,2,2,2,2,2,2,2,2,2,4,248
670 DATA 3,7,13,13,1,1,1,1,1,1,1,1,1,1,1,7
680 DATA 128,128,128,128,128,128,128,128,128,128,128,128
,128,128,128,224
690 DATA 7,8,16,16,0,0,0,0,0,0,1,2,4,8,31
700 DATA 240,8,4,4,4,4,8,16,32,64,128,0,0,0,0,254
710 DATA 15,16,32,0,0,0,0,0,0,0,0,0,32,16,15
720 DATA 240,8,4,2,2,2,6,12,120,12,2,2,2,4,8,240
730 DATA 0,0,0,1,2,4,8,16,32,64,127,0,0,0,0,0
740 DATA 32,96,160,32,32,32,32,32,32,32,252,32,32,32,32
,32
750 DATA 127,64,64,64,64,64,64,127,0,0,0,0,0,0,127
760 DATA 248,0,0,0,0,0,0,240,8,4,4,4,4,4,8,240
770 DATA 0,1,2,4,8,16,32,64,127,64,64,64,64,64,32,31
780 DATA 124,128,0,0,0,0,0,240,8,4,4,4,4,8,240
790 DATA 15,8,0,0,0,0,0,0,0,0,0,0,0,0,0
800 DATA 254,2,2,4,8,16,32,64,128,128,128,128,128,128,1
28,128
810 DATA 31,32,64,64,64,64,64,32,31,32,64,64,64,64,32,31
820 DATA 248,4,2,2,2,2,2,4,248,4,2,2,2,2,4,248
830 DATA 31,32,64,64,64,64,32,31,0,0,0,0,0,0,32,63
840 DATA 248,4,2,2,2,2,2,254,2,2,2,2,2,4,8,240
```



サンプルプログラム-3「MSXリズムボックス」

```
100 '---------------
110 '
120 '  RHYTHM-BOX
130 '
140 '--------------
150 '
160 L=100
170 CLS:PRINT "(F1)--> SLOW"
180 PRINT "(F2)--> FAST"
190 PRINT "(F3)--> MANUAL"
200 KEY (1) ON:KEY (2) ON:KEY (3) ON
210 ON KEY GOSUB 480,500,530
220 '
230 '--- AUTO MODE ---
240 '
250 FOR I=0 TO 13
260 READ A:SOUND I,A
270 NEXT I
280 FOR I=1 TO L+42
290 NEXT I
300 SOUND 13,9
310 FOR I=1 TO L-14
320 NEXT I
330 FOR I=0 TO 13
340 READ A:SOUND I,A
350 NEXT I
360 FOR I=1 TO L-20
370 NEXT I
380 RESTORE
390 FOR I=0 TO 13
400 READ A:SOUND I,A
410 NEXT I
420 FOR I=1 TO L-14
430 NEXT I
440 RESTORE:GOTO 250
450 DATA 20,9,0,0,0,0,28,53,16,0,0,23,9,9
460 DATA 0,0,0,0,0,0,5,53,16,0,0,23,9,9
470 '--- SLOW ---
480 L=L+2:RETURN
490 '--- FAST ---
500 IF L=<1 THEN L=1:RETURN
510 L=L-2:RETURN
520 '
```

```
530 '--- MANUAL MODE ---
540 '
550 KEY (3) OFF
560 A=STICK(0):IF A=0 THEN 560
570 IF (A=3) OR (A=7) OR (A=1) THEN 590
580 GOTO 560
590 IF A=1 THEN KEY (3) ON:RETURN
600 RESTORE
610 IF A=3 THEN FOR I=0 TO 13:READ A:SOUND I,A:NEXT I:
GOTO 560
620 RESTORE 460
630 FOR I=0 TO 13:READ A:SQUND I,A:NEXT I:GOTO 560
```

ボクらが日頃おなじみの音に，救急車の「ピーポーピーポー」というサイレンの音がある．次のプログラムは，PSGを使ってあの音を真似してみるものだ．高い音と低い音を２つ用意して切り換えている．

もう少しリアリティを出したいという人は，音量を少しずつ上げ下げしていって，だんだん近づいて遠ざかっていく感じを再現すれば良いだろう．同時に音の高さの方も一番近付いたときから後，少し下げてやれば，ドプラー効果のように聞こえて，ほとんど本物そっくりになる．ちょっとしたプログラムの改良で実現できるはずだ．試してみよう．

☆

```
100 RESTORE
110 FOR I=0 TO 13:READ A
120 SOUND I,A:NEXT I
130 FOR W=1 TO 220:NEXT W
140 FOR I=0 TO 13:READ A
150 SOUND I,A:NEXT I
160 FOR W=1 TO 210:NEXT W
170 GOTO 100
180 '---ｻｲﾚﾝ HIGH
190 DATA 98,0,99,0,100,0,0,56,10,10,10,0,0,0
200 '--- ｻｲﾚﾝ LOW
210 DATA 123,0,125,0,128,0,0,56,10,10,10,0,0,0
```

SOUND PACK

サウンド・データ集

さて，最後に本書の締めくくりとして，「マイコン・サウンド・パック」の名にふさわしく，PSGのサウンド・データ集を載せておく．

ここに掲げた30種の音のデータは，いずれもサウンド・エディタを使ってあれこれ試しているうちに生まれたものだ．中には「ン？」と思うような強引な解釈（！）により名前を付けたものもあるが，できるだけ具体的に何かのイメージを呼び起こすような音を選んだつもりである．もっともPSGは「PSGの音」としか言いようがないような非現実的な音出しが一番得意なのだけれども………．

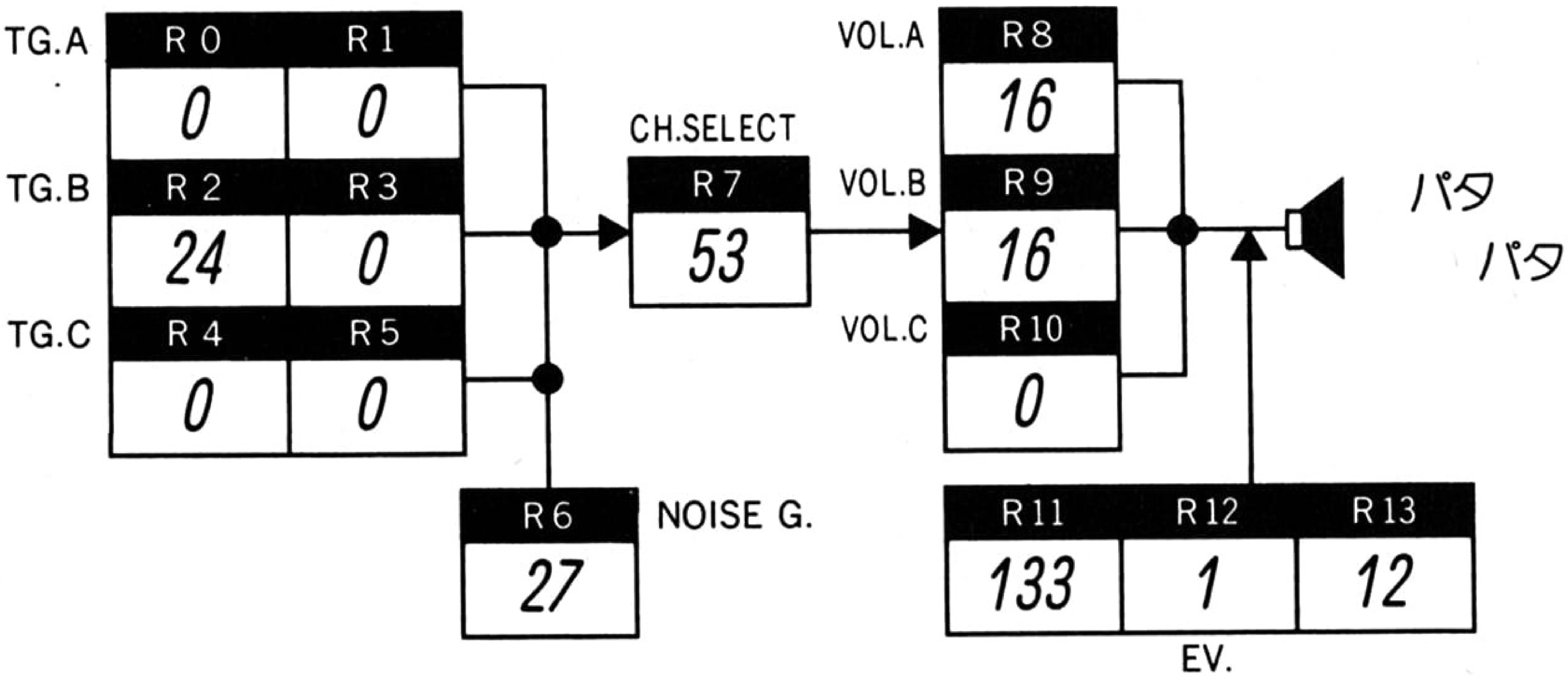
ゲームなど他のプログラムに使いたい場合には，次のリストのようにDATA文＋サウンド・データ書き込みルーチンの形にして，適当な行番号に配置してやればよい．5章のサンプル・プログラムを参考にして，画面と音との組み合わせ方などを身につけよう．

ちょっとした音を加えるだけで，演出効果は格段に高まるはずだ．

```
100 RESTORE 1010
110 FOR I=0 TO 13
120 READ SD:SOUND I,SD
130 NEXT I
1000 '--- XX ノ オト ---
1010 DATA 254,0,202,0,170,0,0,56,16,16,16,10,0,14
```

ヘリコプターの音 1

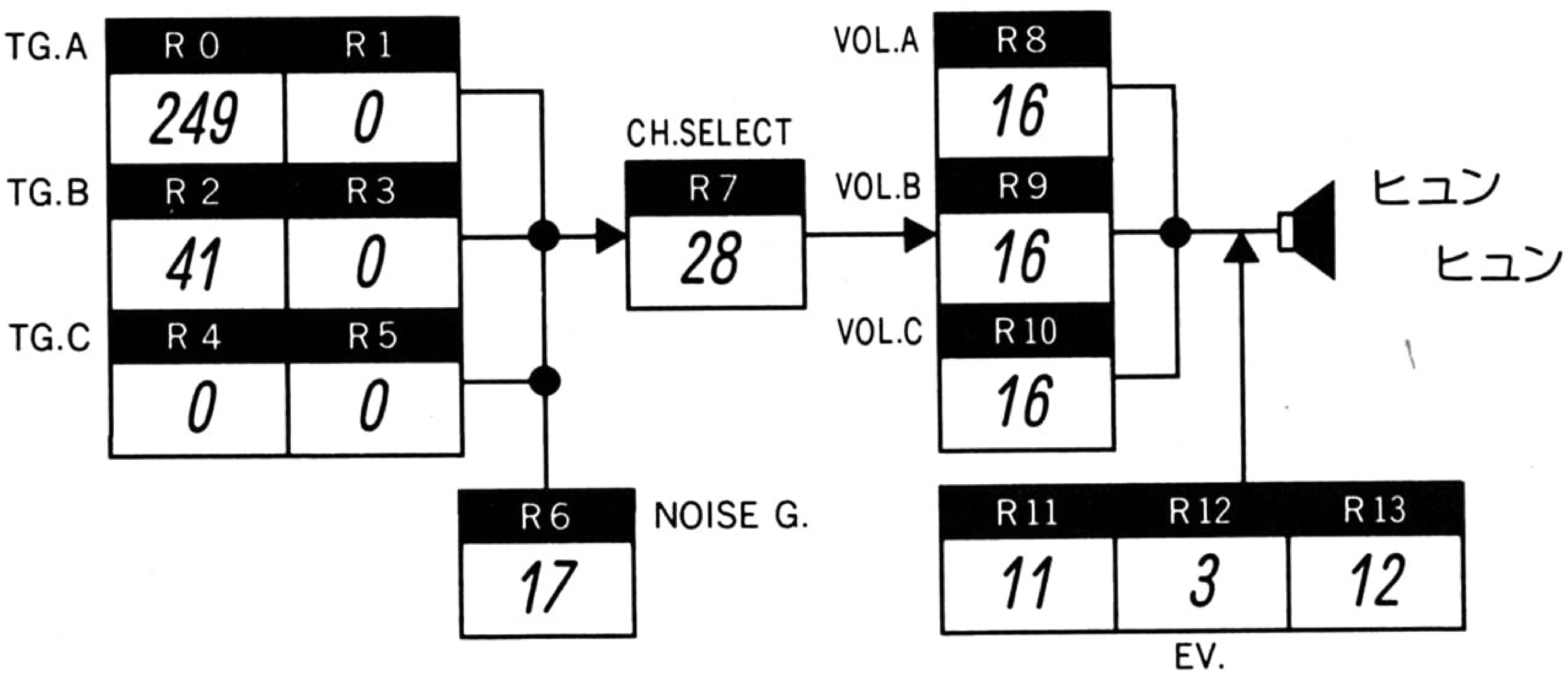
あの「地獄の黙示録」でおなじみの，ジェット・ヘリ戦闘機の音．エンベロープの周期を変えると，ローターの回転速度が変化する感じが出る．




```
DATA 0,0,24,0,0,0,27,53,16,16,0,133,1,12
```

ヘリコプターの音 2

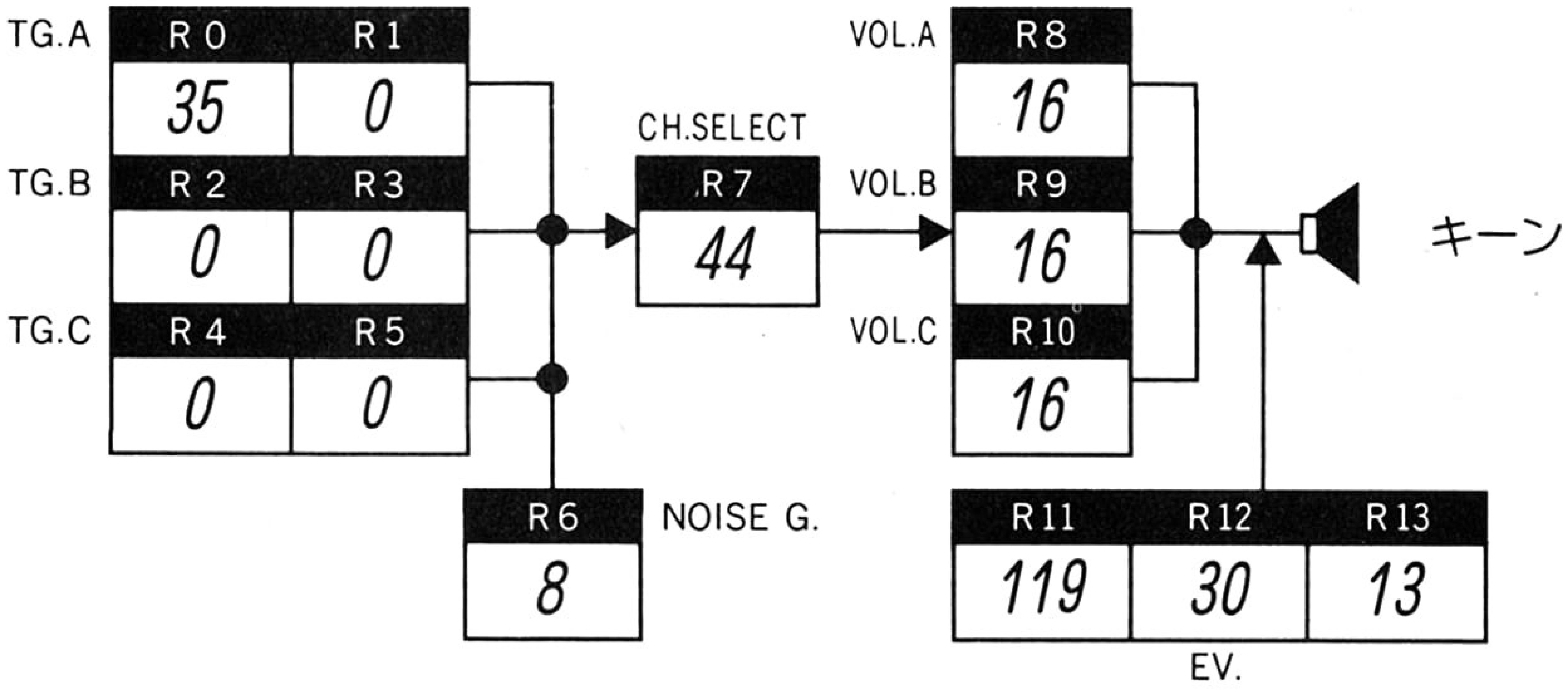
これも同じようなヘリの音だが，CH-CのノイズにCH-A，CH-Bから高めのトーンを加えて固い感じの音にしてある．離陸を前にして待機中といったところ．




```
DATA 249,0,41,0,0,0,17,28,16,16,16,11,3,12
```

ジャンボ・ジェット機内音

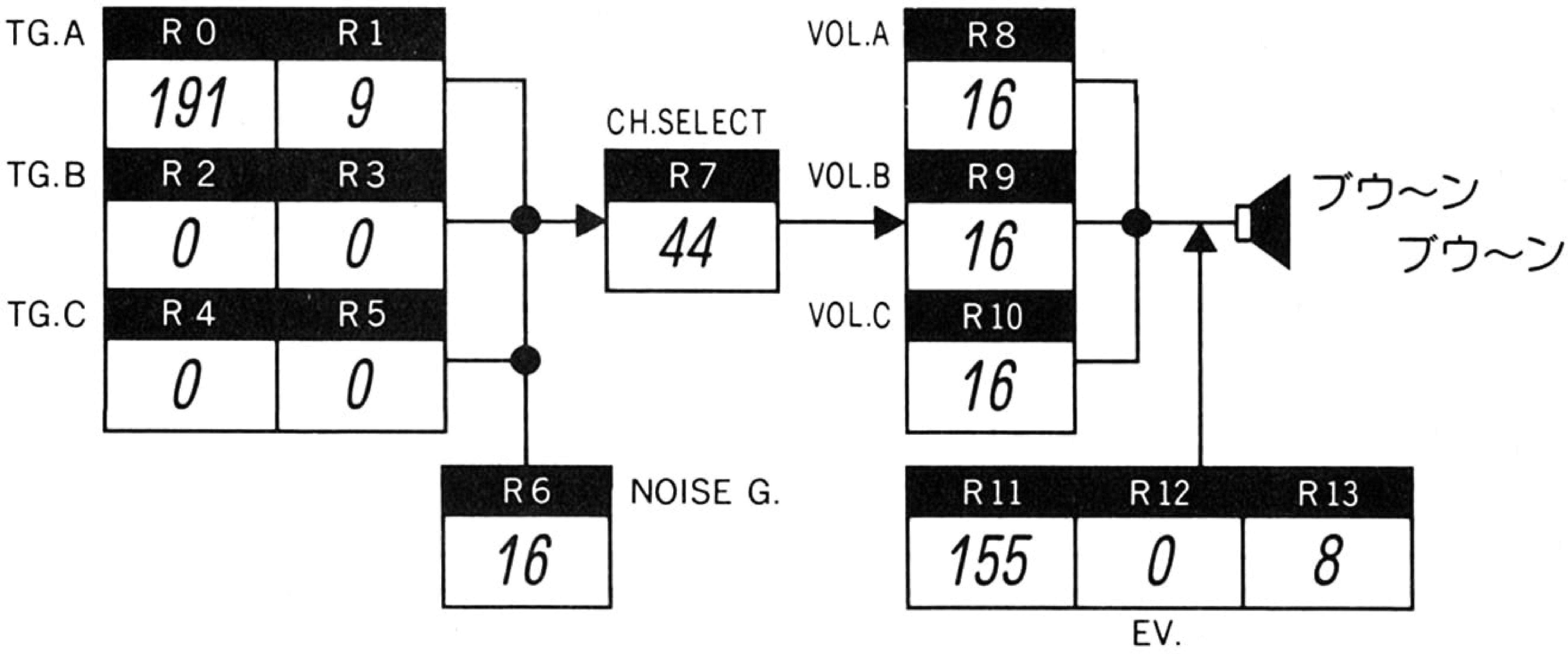
ジャンボ・ジェット機の客席内の雰囲気を出してみた．巡航速度で安定飛行中．




```
DATA 35,0,0,0,0,0,8,44,16,16,16,119,30,13
```

プロペラ機の音

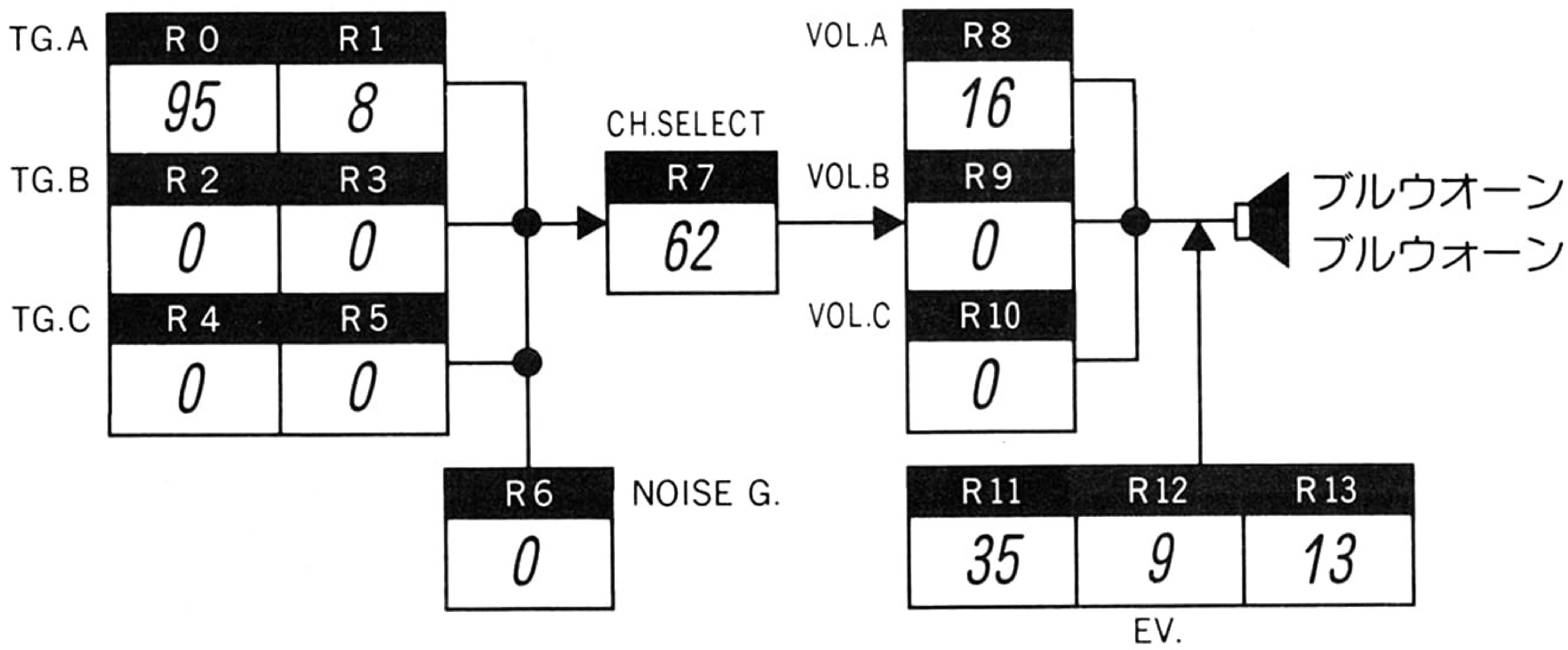
今度は懐しいプロペラ機の音だ．CH-Aの低いトーンとCH-Bのノイズを混ぜてある．このようにエンベロープ周期を短くすると，面白い効果が生まれる．




```
DATA 191,9,0,0,0,0,16,44,16,16,16,155,0,8
```

自動車のエンジン音

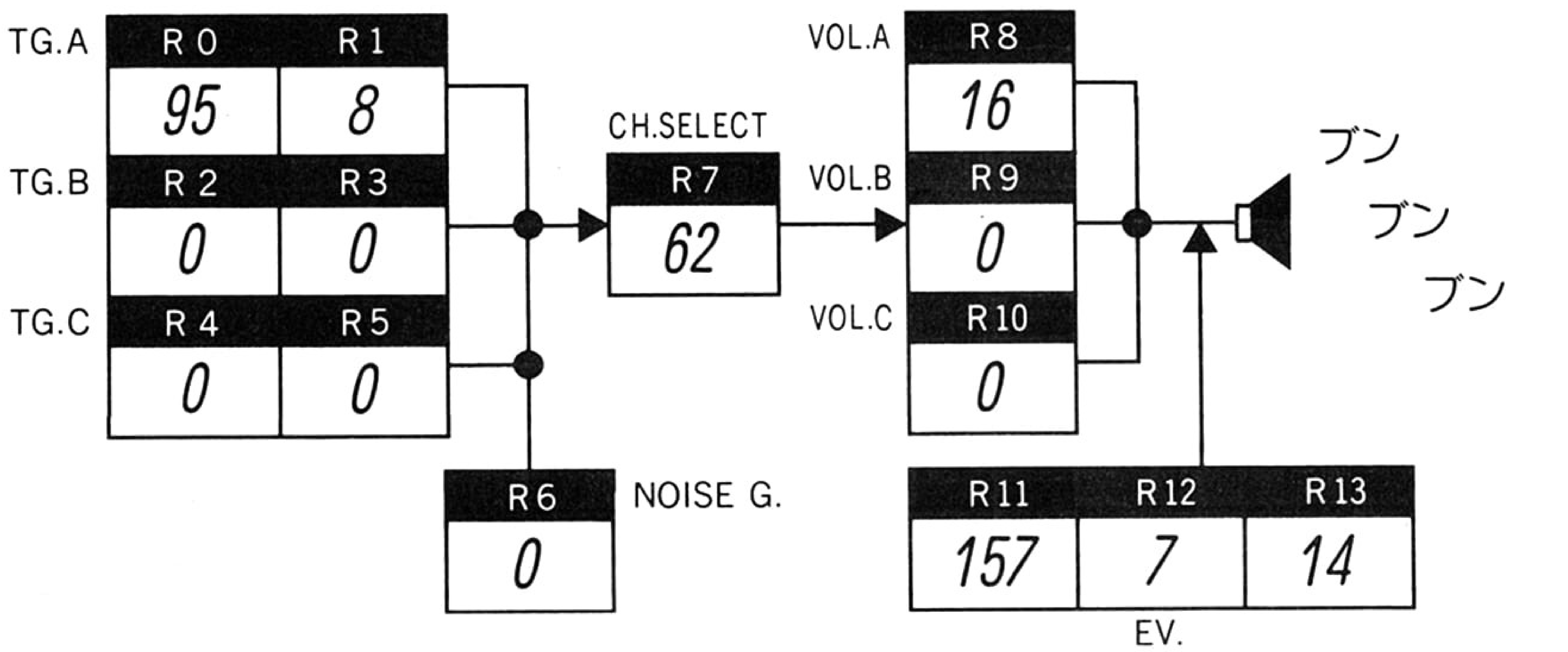
この音は一見（聞？）ノイズのようだけれども，実はCH-Aからのトーンだけで作り出している．乗用車の走行音といったところ．ドライヴ・ゲームなどに使おう．




```
DATA 95,8,0,0,0,0,0,62,16,0,0,35,9,13
```

プロペラ機エンジン不調

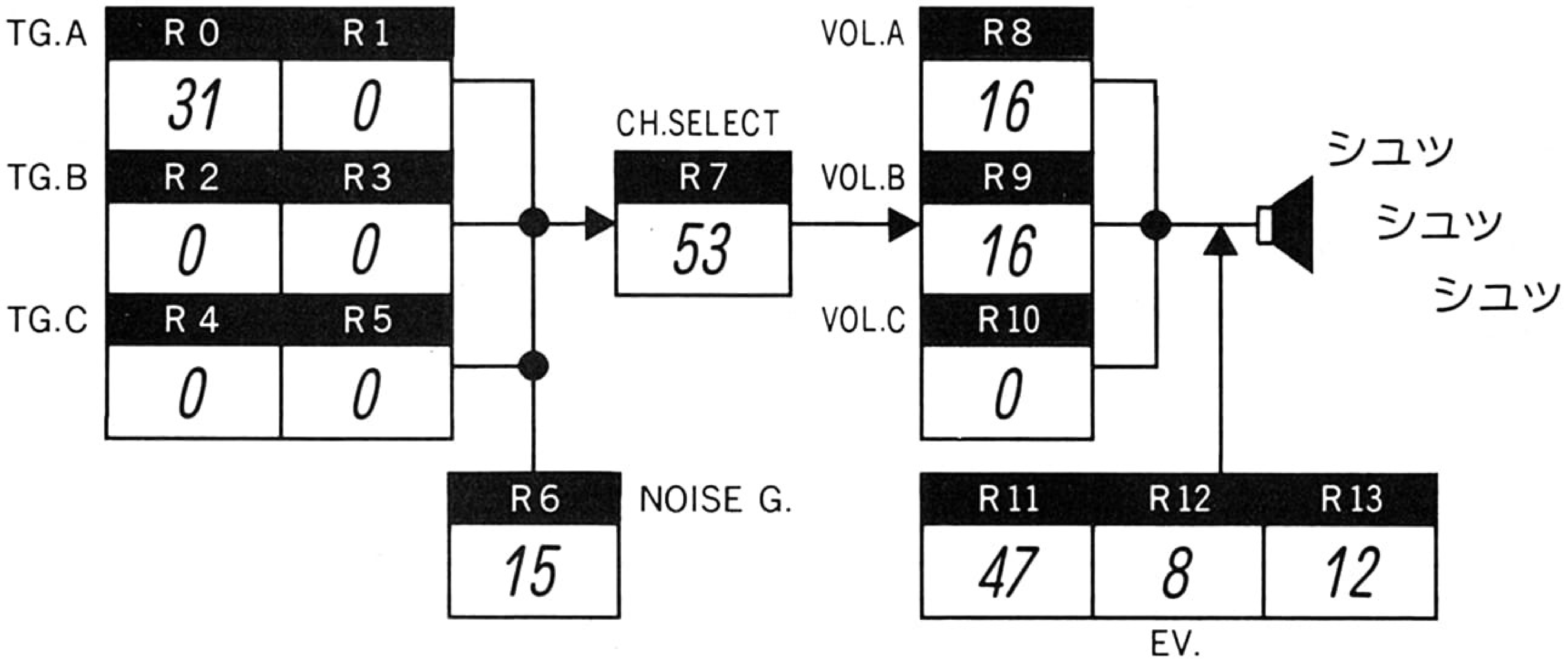
暴風雨に立ち向かうセスナ機が，風に吹かれて大揺れしている……というのは，ちょっと想像力が豊かすぎるかな？この音もCH-Aのトーンだけで作られている．




```
DATA 95,8,0,0,0,0,0,62,16,0,0,157,7,14
```

SLの音

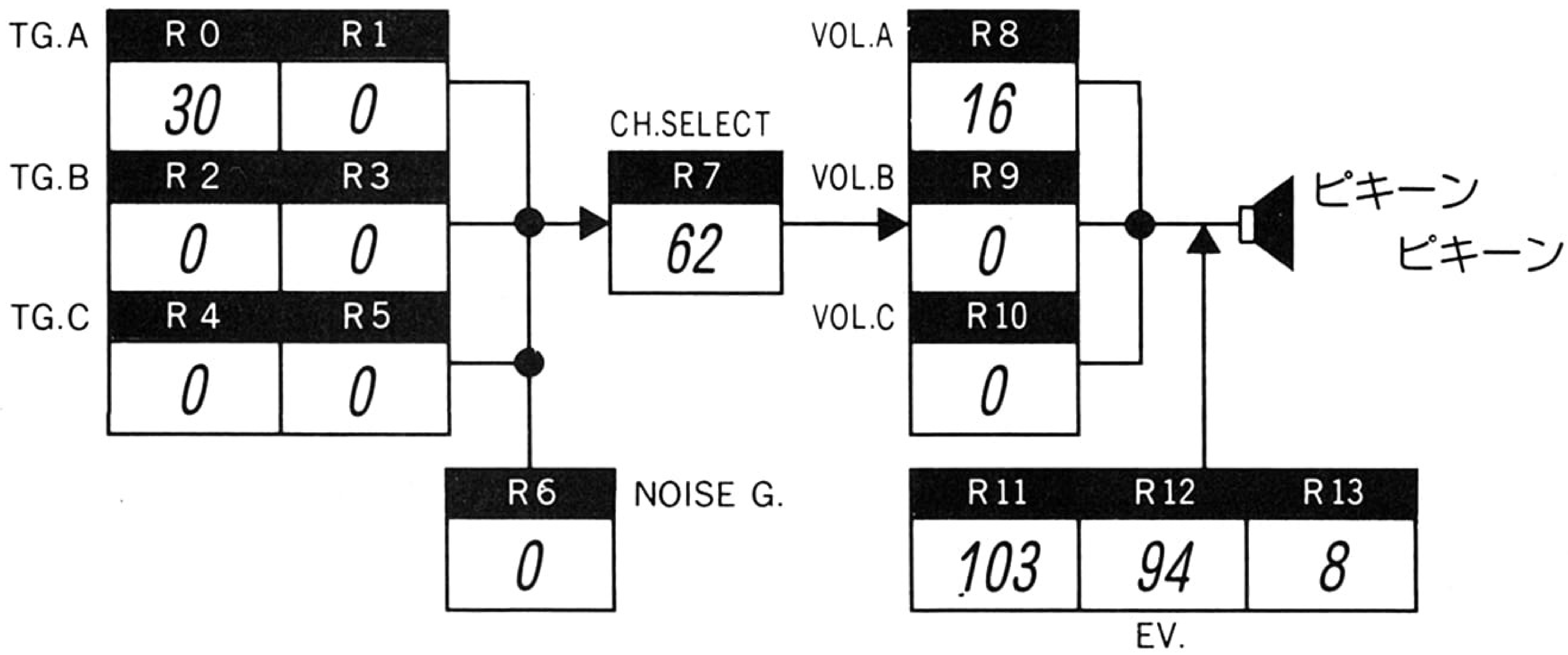
SL(蒸気機関車)の音だ．FOR～NEXT文を使ってレジスタ12=30ぐらいから少しずつエンベロープ周期を短くしていくと，始動から走り出すまでの感じが出せる．




```
DATA 31,0,0,0,0,0,15,53,16,16,0,47,8,12
```

潜水艦のソナー

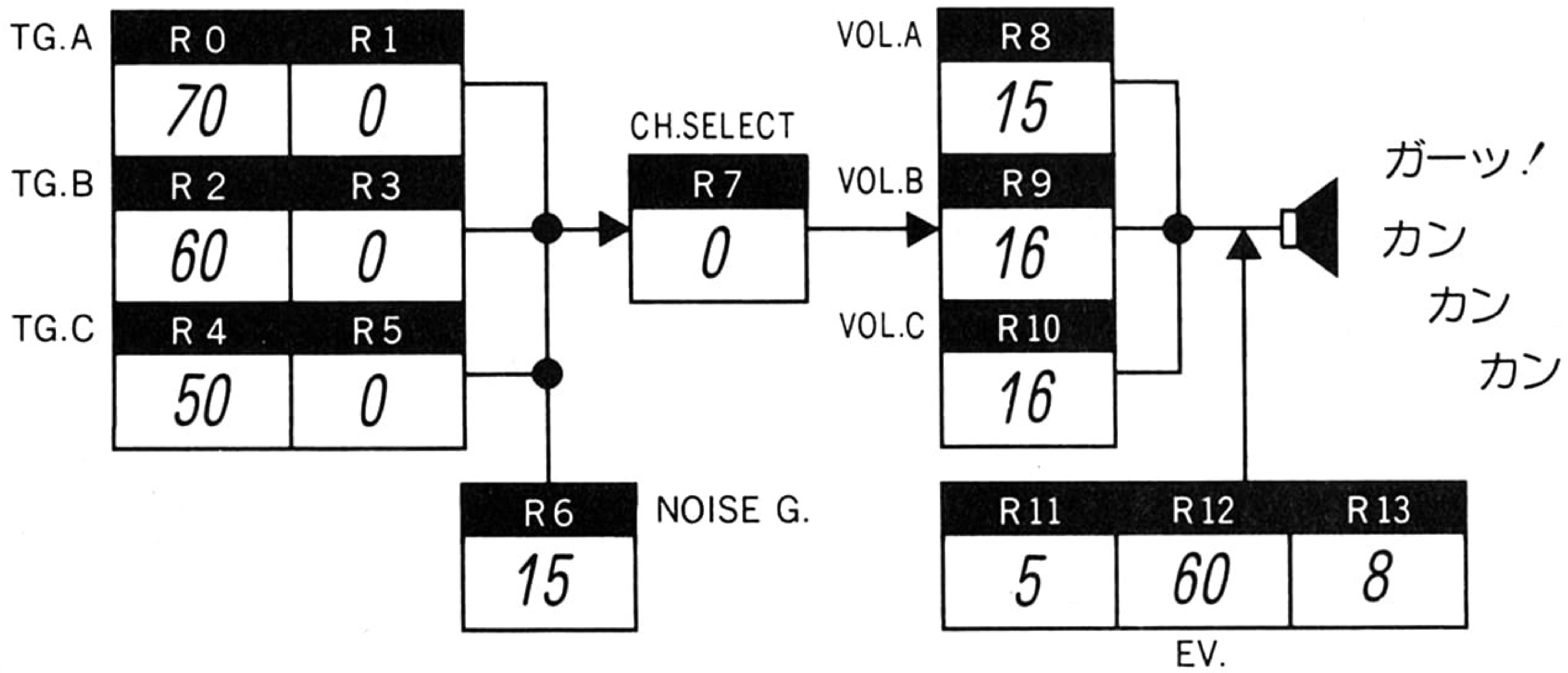
潜水艦のソナーの音のつもり．深海を舞台にしたゲームには欠かせない小道具だ．




```
DATA 30,0,0,0,0,0,0,62,16,0,0,103,94,8
```

工事現場の音

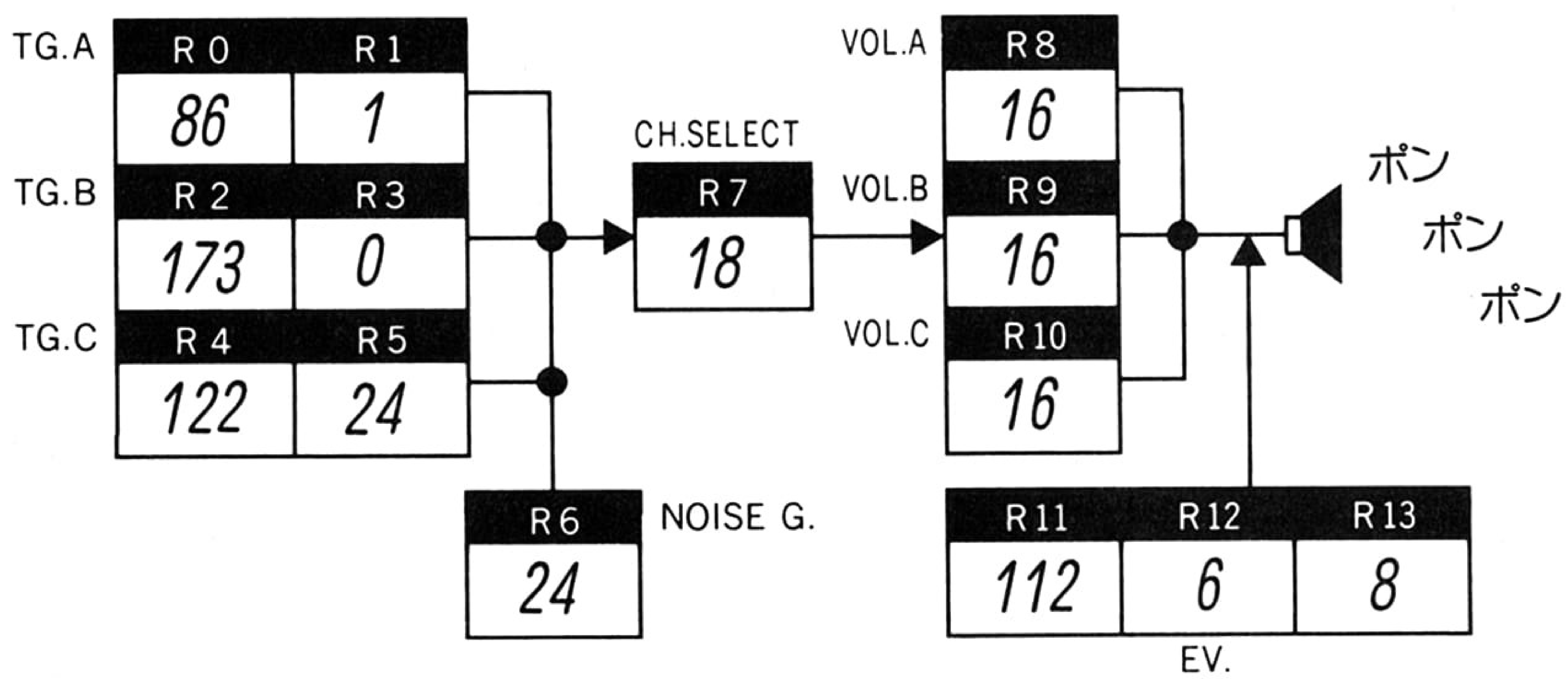
ビルの建設現場とか，工場の雰囲気を出したつもり．この音の場合はCH-BとCH-Cで「カン・カン」という金属音を出し，CH-Aで連続したノイズを出している．




```
DATA 70,0,60,0,50,0,15,0,15,16,16,5,60,8
```

ディーゼル発動機の音

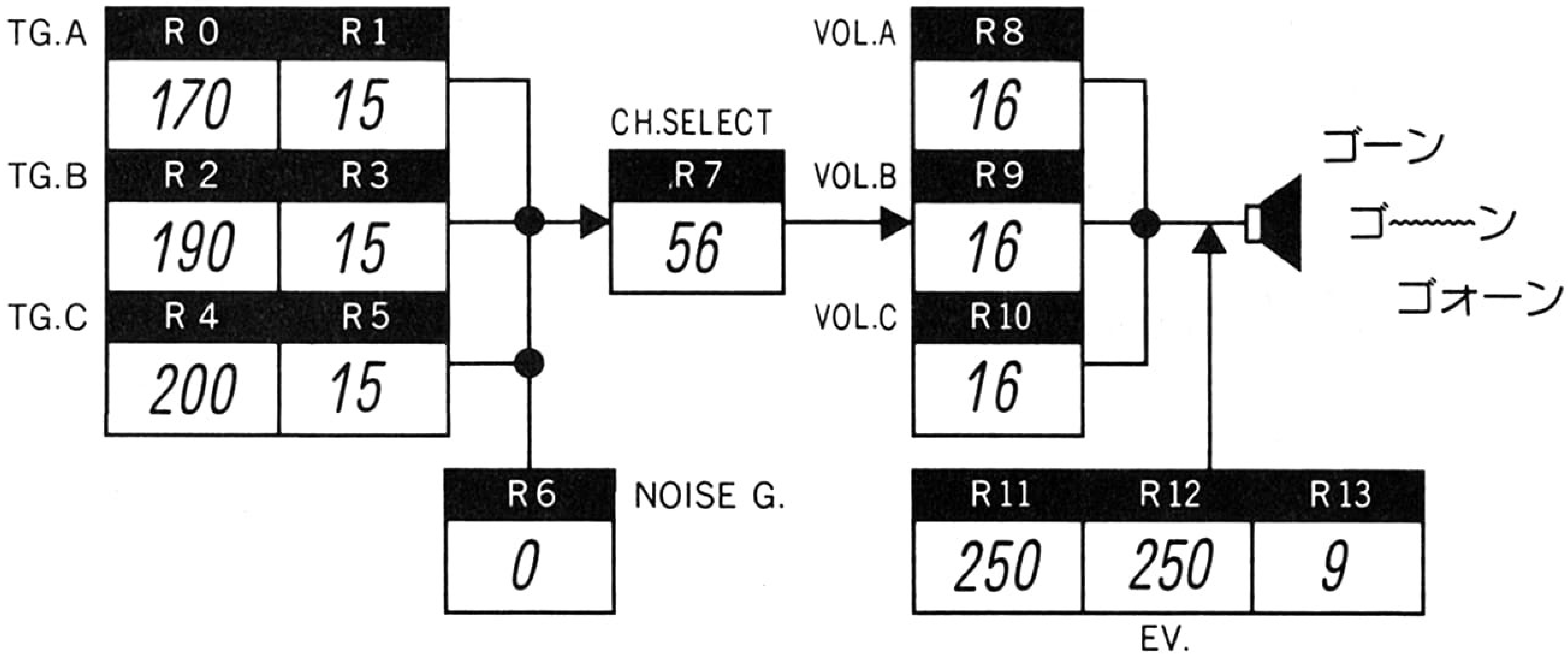
ディーゼル・エンジンというのは，バスや小型の船なんかに使われている．エンベロープの周期を上げると高速回転のエンジン音に聞こえる．




```
DATA 86,1,173,0,122,24,24,18,16,16,16,112,6,8
```

お寺の鐘の音

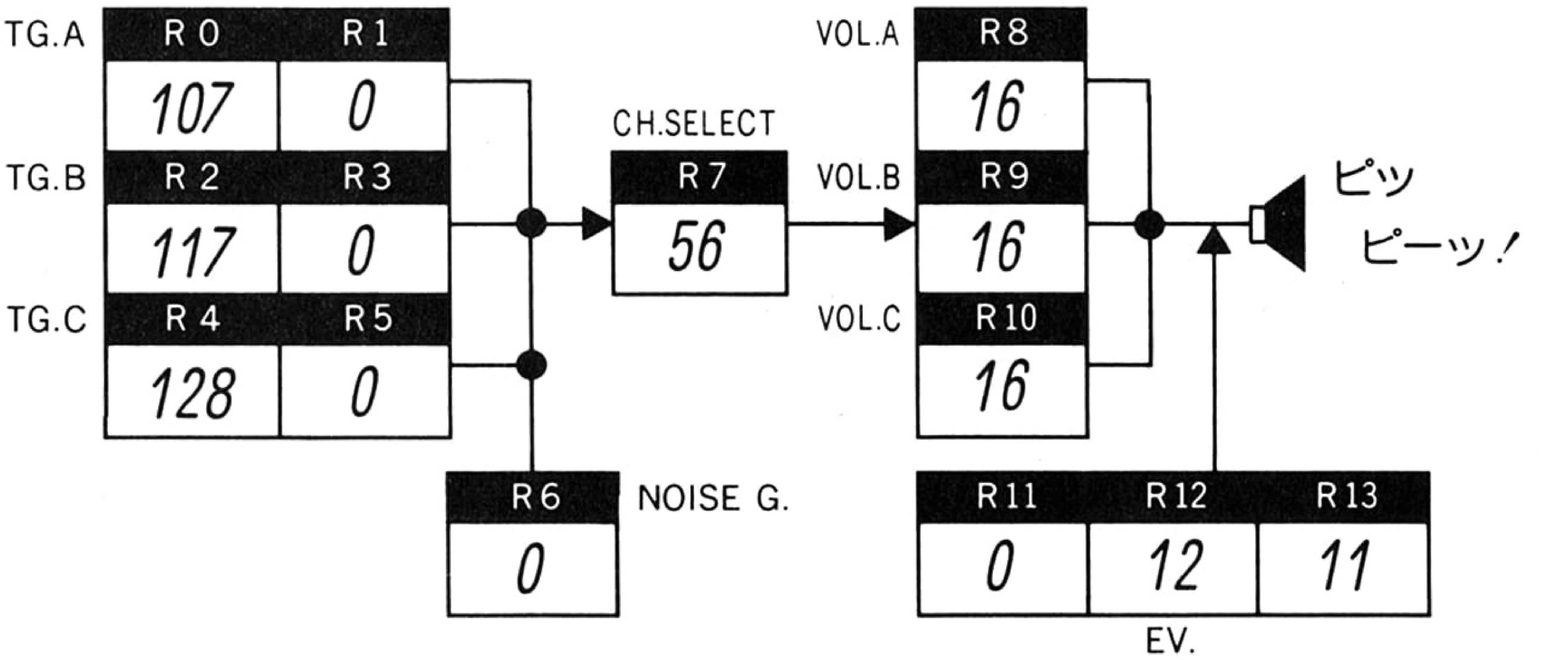
198X年の暮も押し詰まってきた．静かな夜空に除夜の鐘だけが鳴り渡っていく．鐘の音にしては少し暖かみに欠けるが，まあ我慢してほしい．




```
DATA 170,15,190,15,200,15,0,56,16,16,16,250,250,9
```

汽笛の音

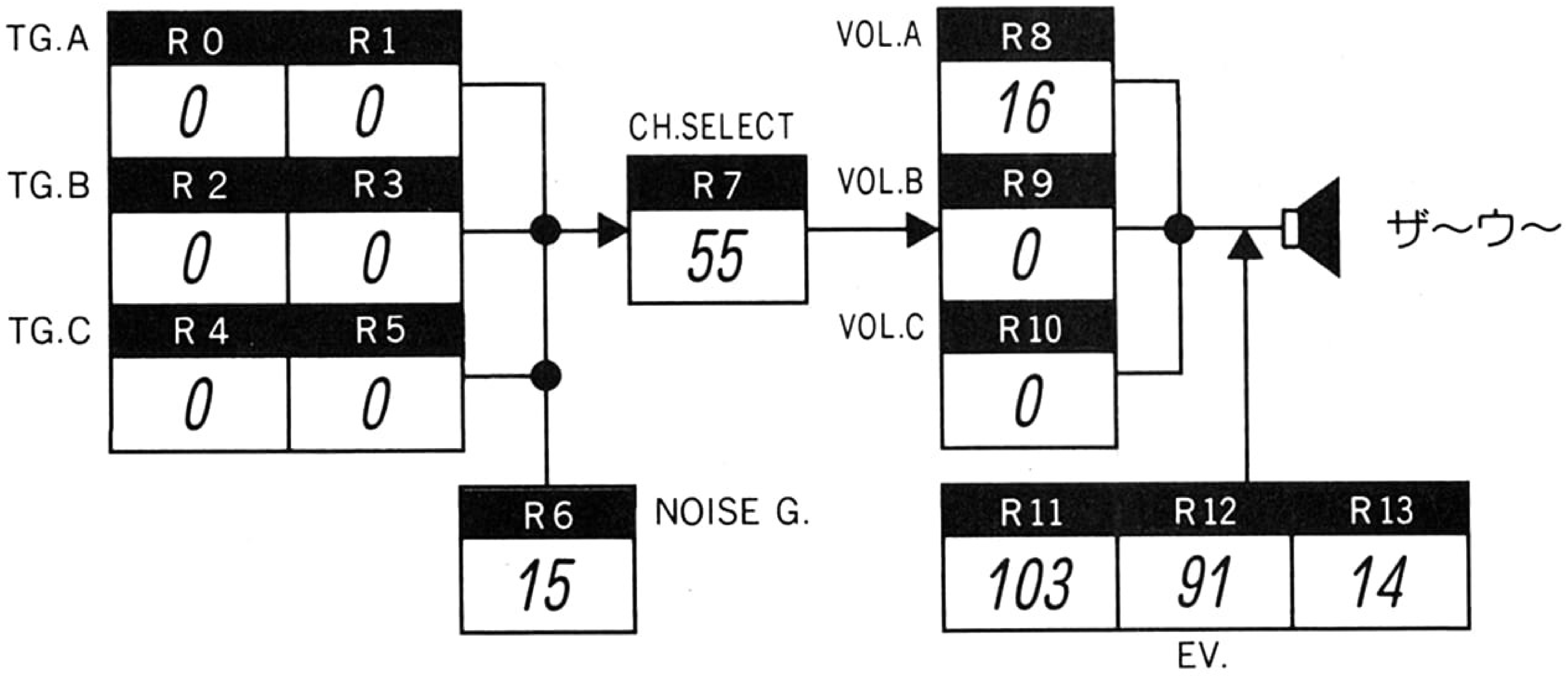
汽車の汽笛の音だ．上野発の夜行列車，なんちゃって．このままだと音が続きっ放しなので，実際に使うときには一定時間をおいてレジスタ7に63を書きこむようにする．




```
DATA 107,0,117,0,128,0,0,56,16,16,16,0,12,11
```

波の音

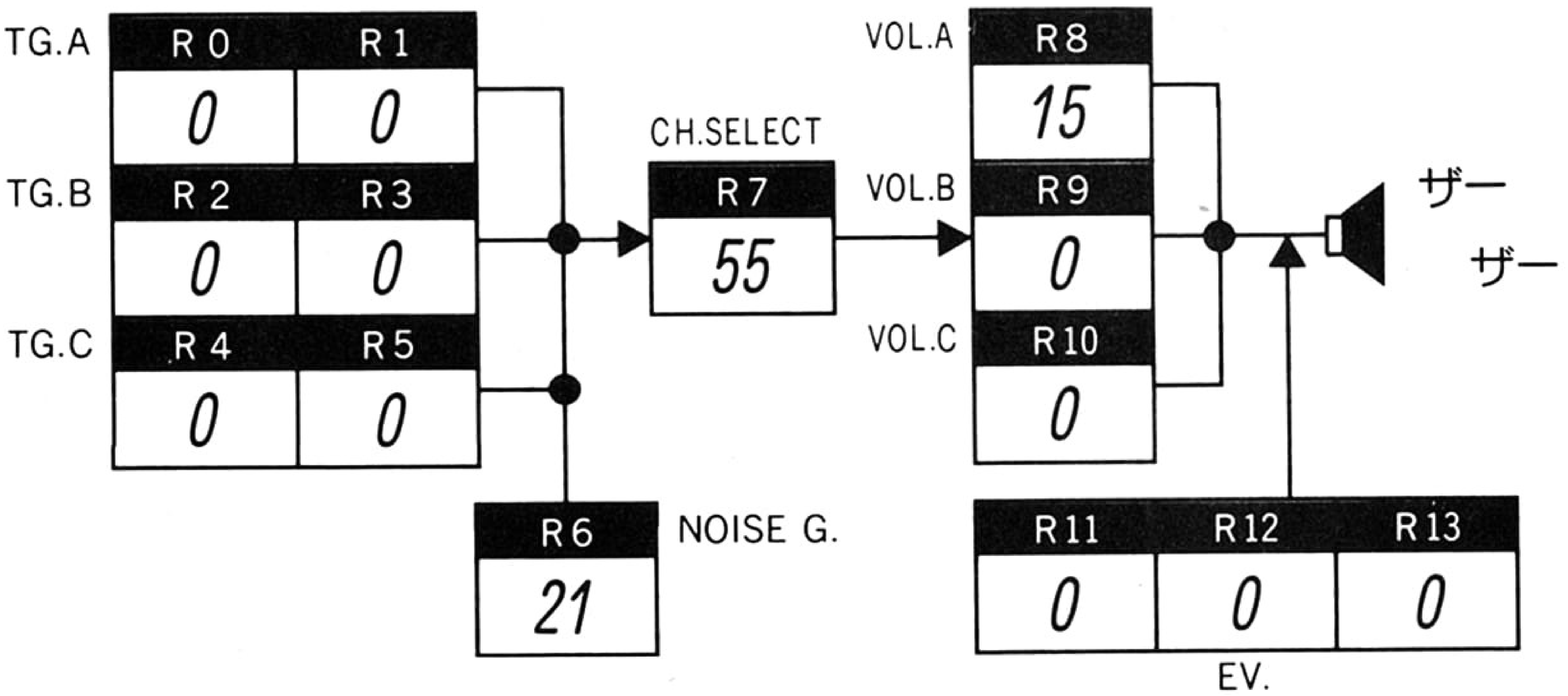
１章でも紹介した波の音だ．環境ソフトとしては一番の出きばえだと思うが，どうかな?クーラーの効かない暑い夜でもこの音を聞くとさわやかな気分になるはず．




```
DATA 0,0,0,0,0,0,15,55,16,0,0,103,91,14
```

滝の音

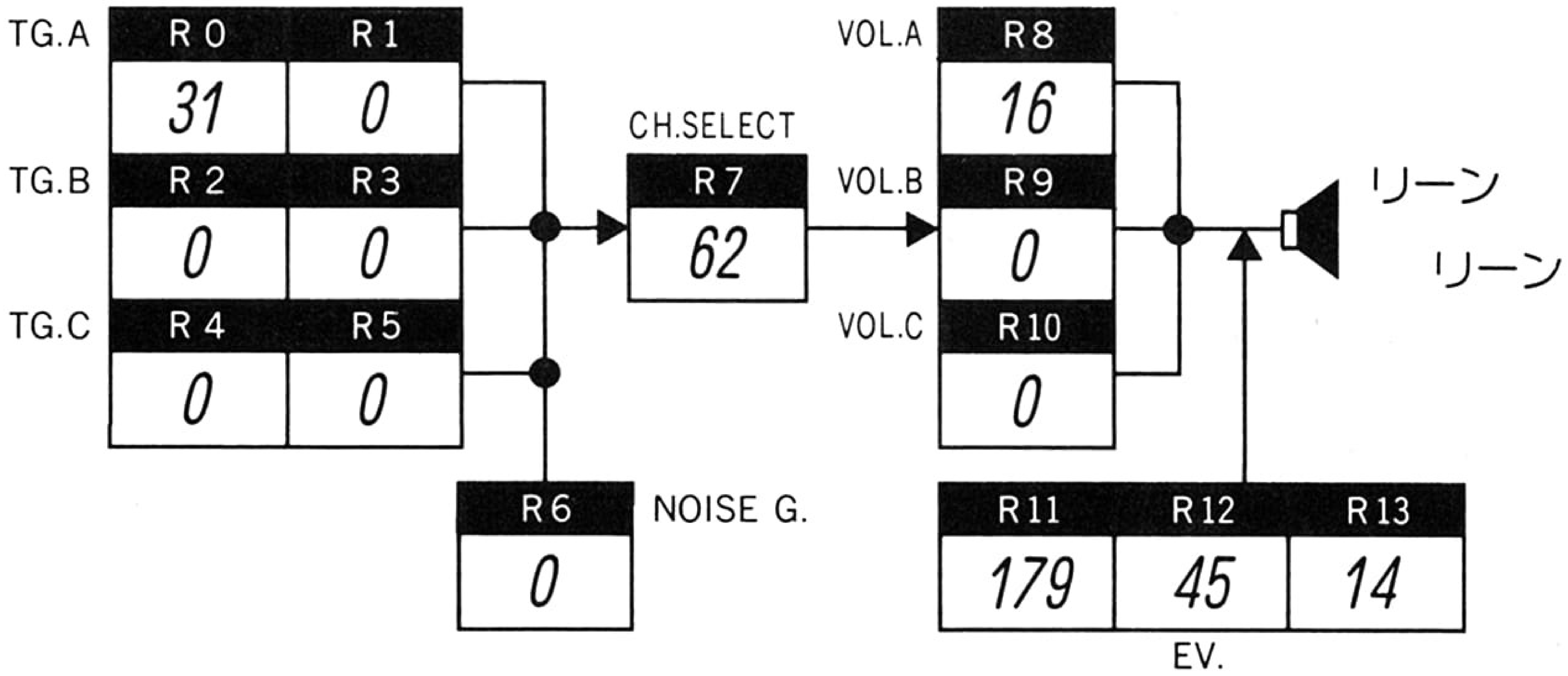
環境ソフト第２弾は，滝の音．目を閉じると大きな滝や放流中のダムのそばにいるような気がしてくる．音は何の変哲もないただのノイズだ．




```
DATA 0,0,0,0,0,0,21,55,15,0,0,0,0,0
```

スズムシの鳴き声

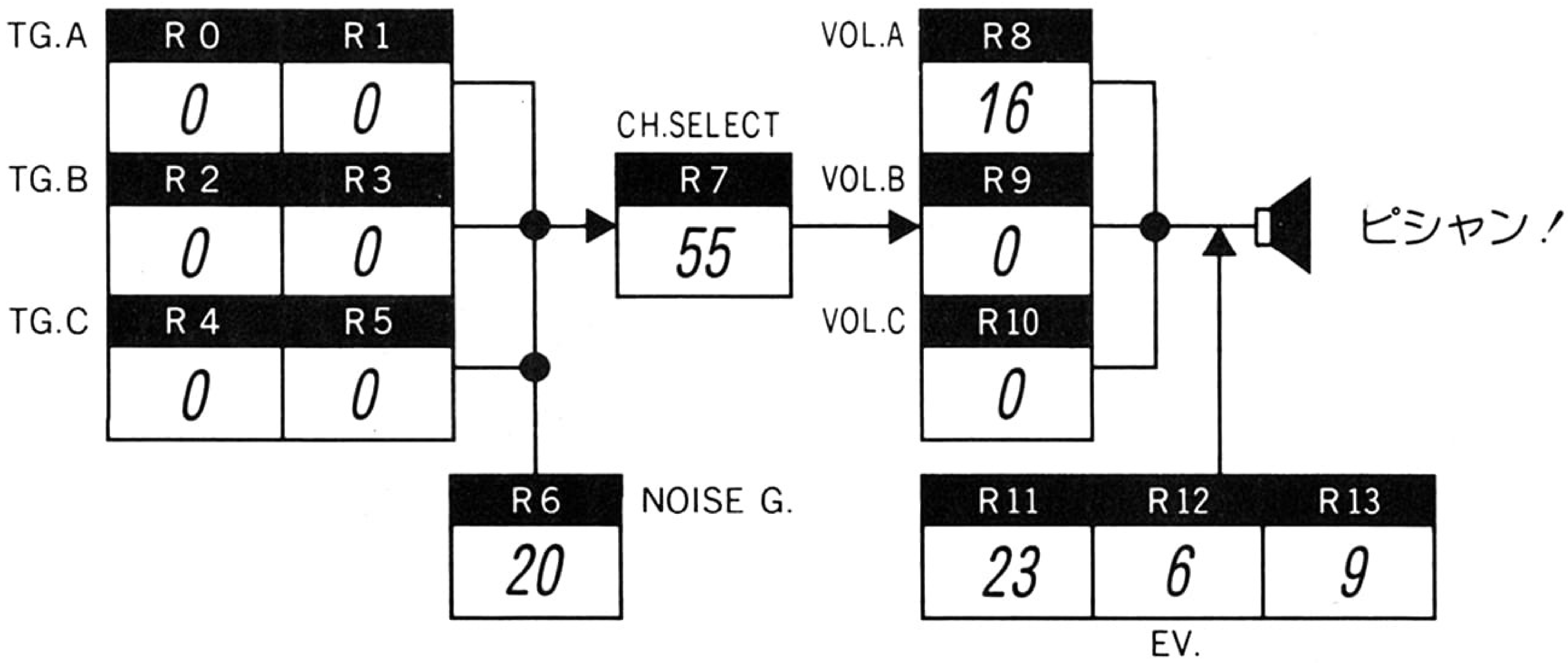
「秋の夜長を鳴き通す」日本の秋だ，キンチョーだ．本物よりはやや音色がキンキンしているが，我慢しよう．まったく発想を変えてUFOの音に使っても良いかも．




```
DATA 31,0,0,0,0,0,0,62,16,0,0,179,45,14
```

ビンタの音

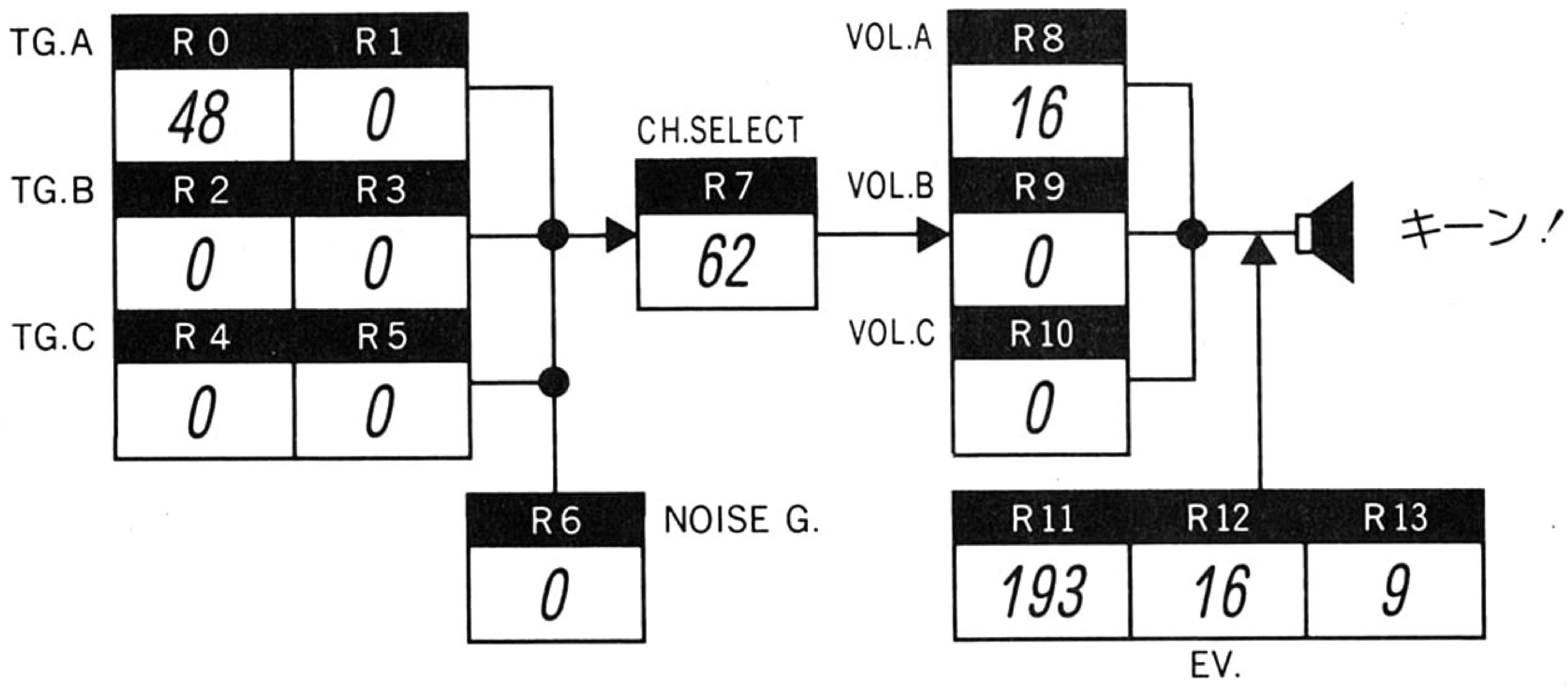
あこがれの彼女に平手打ちをくらってしまった……ううっ涙．そんなに怒らなくたっていいじゃん．このようにいきなり途切れる音は，心理的に拒絶感を与えるのだ．




```
DATA 0,0,0,0,0,0,20,55,16,0,0,23,6,9
```

金属バットの音

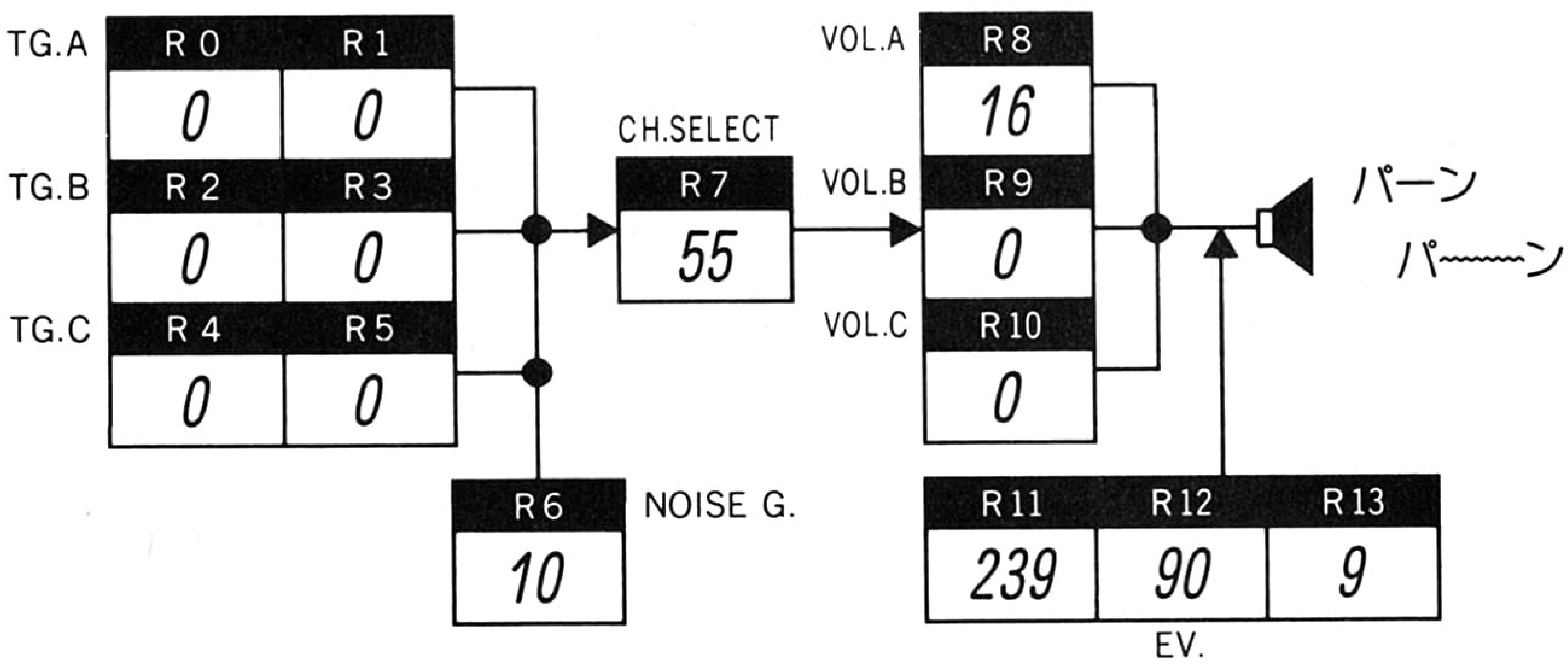
「カキーン」というジャストミートの音には程遠いが，ファウルチップぐらいの感じは出ているのでは？なお，戸塚ヨットスクール等とは一切関係ありません．




```
DATA 48,0,0,0,0,0,0,62,16,0,0,193,16,9
```

銃声

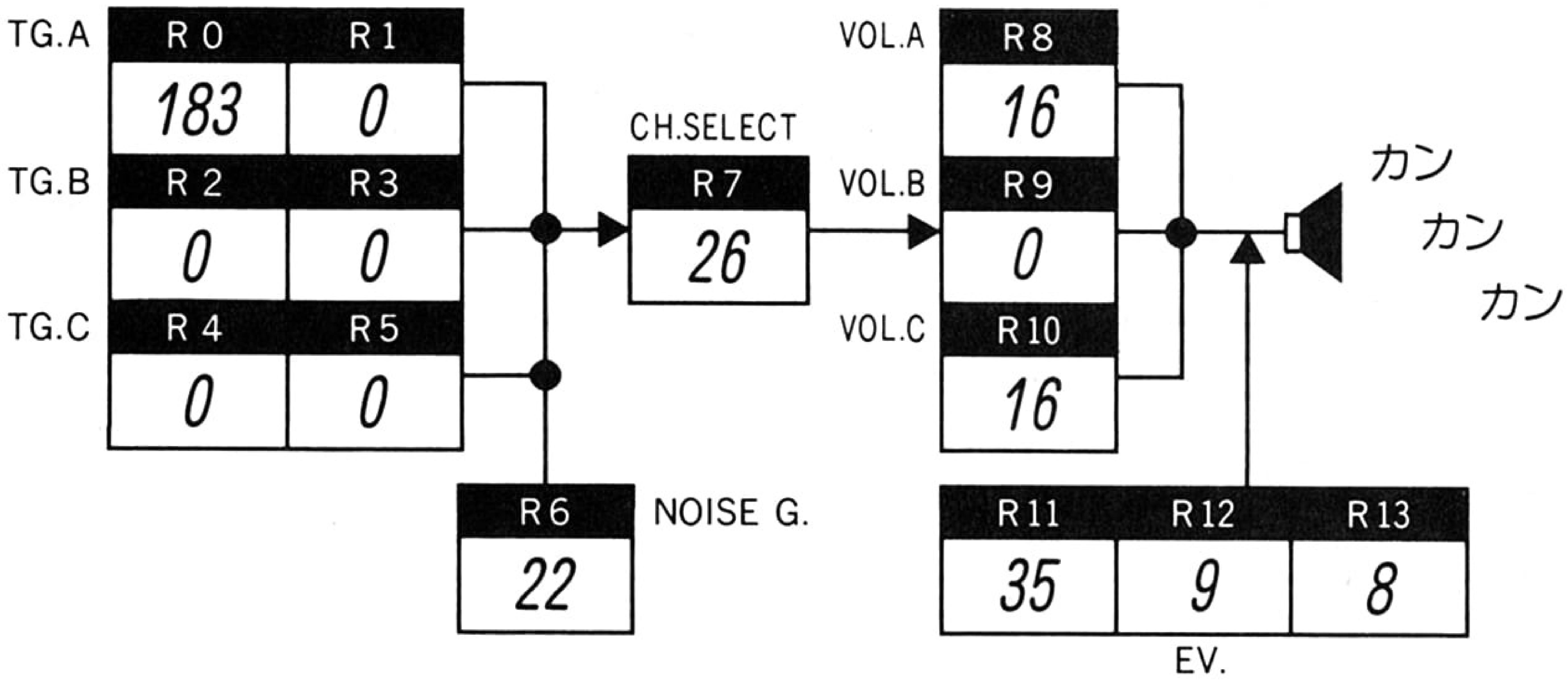
広大な雪の原野にとどろく銃声一発．そのとき最後のオオカミ王は真赤な血に染まって倒れたのであった……．かためのノイズに長いエンベロープをかけて処理している．




```
DATA 0,0,0,0,0,0,10,55,16,0,0,239,90,9
```

踏切り

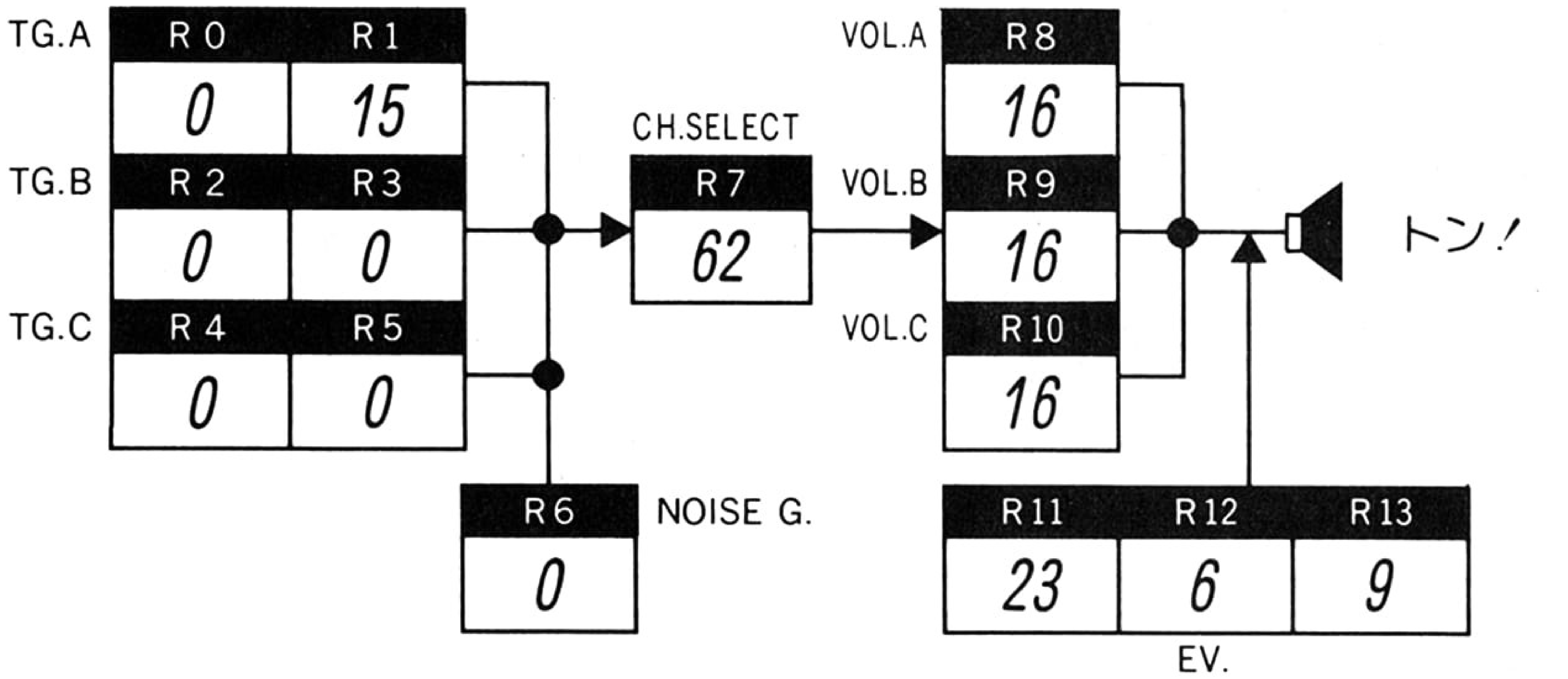
電車の踏切りの音のつもりだ．どちらかというと昔懐しいベル式の警報器というよりは，私鉄あたりの電子式の踏切りの音に近いのでは？




```
DATA 183,0,0,0,0,0,22,26,16,0 ,16,35,9,8
```

ドラムの音 1

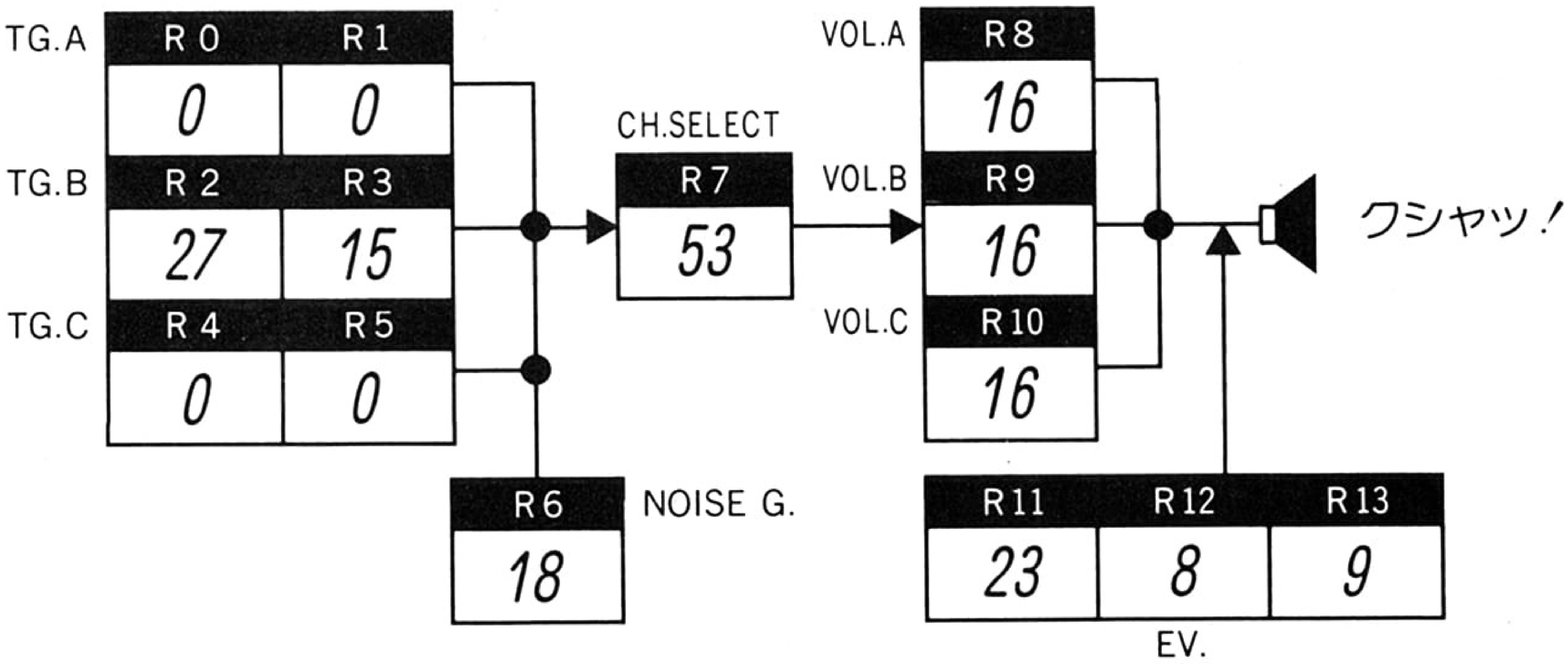
PSGで一番出しやすい楽器の音というと，やっぱり打楽器の音なのだ．この音は重い感じで，バスドラムかタムタムといったところだろう．




```
DATA 0,15,0,0,0,0,0,62,16,16,16,23,6,9
```

ドラムの音 2

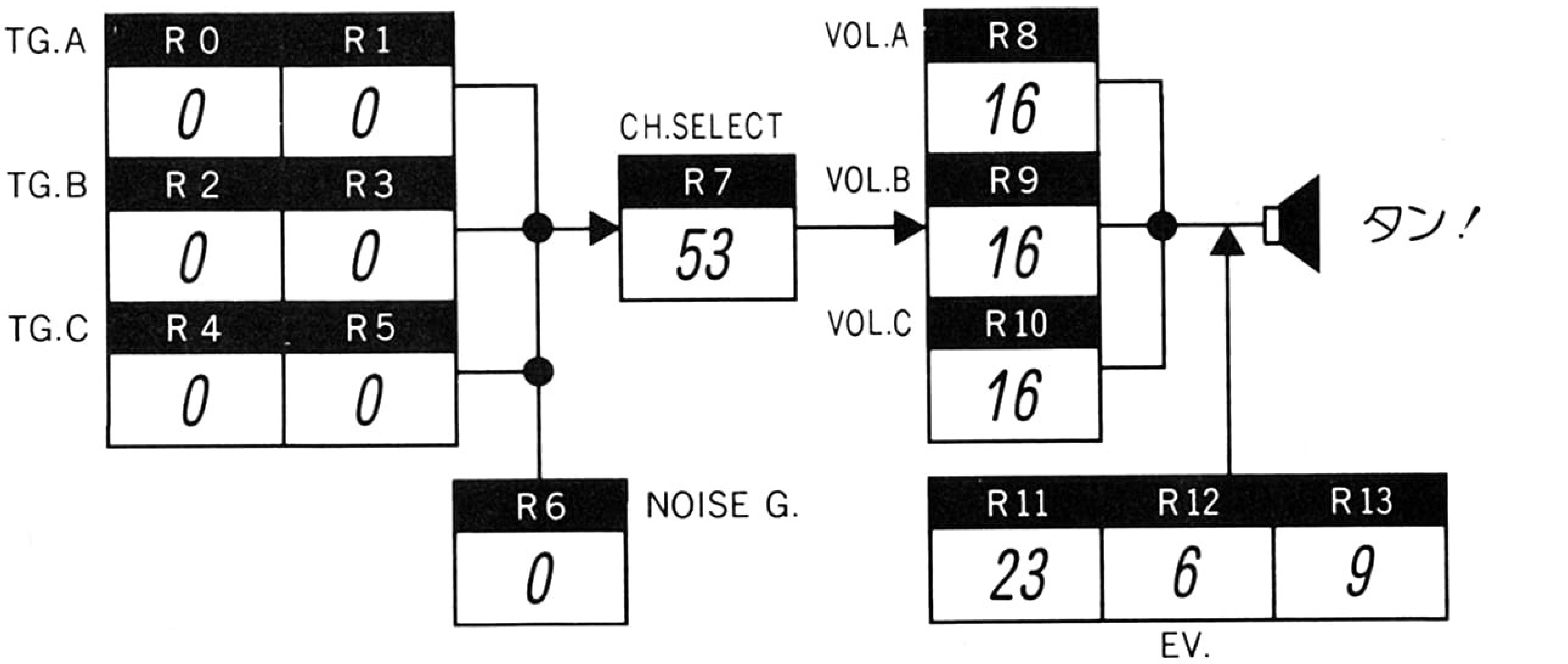
YMOなどがよく使う，シンセドラムの音に聞こえないかな？ひずんだ感じの音は，クラッシシュ・シンバル(シンバルを裏返しにしたもの)のようでもある．




```
DATA 0,0,27,15,0,0,18,53,16,16,16,23,8,9
```

ドラムの音 3

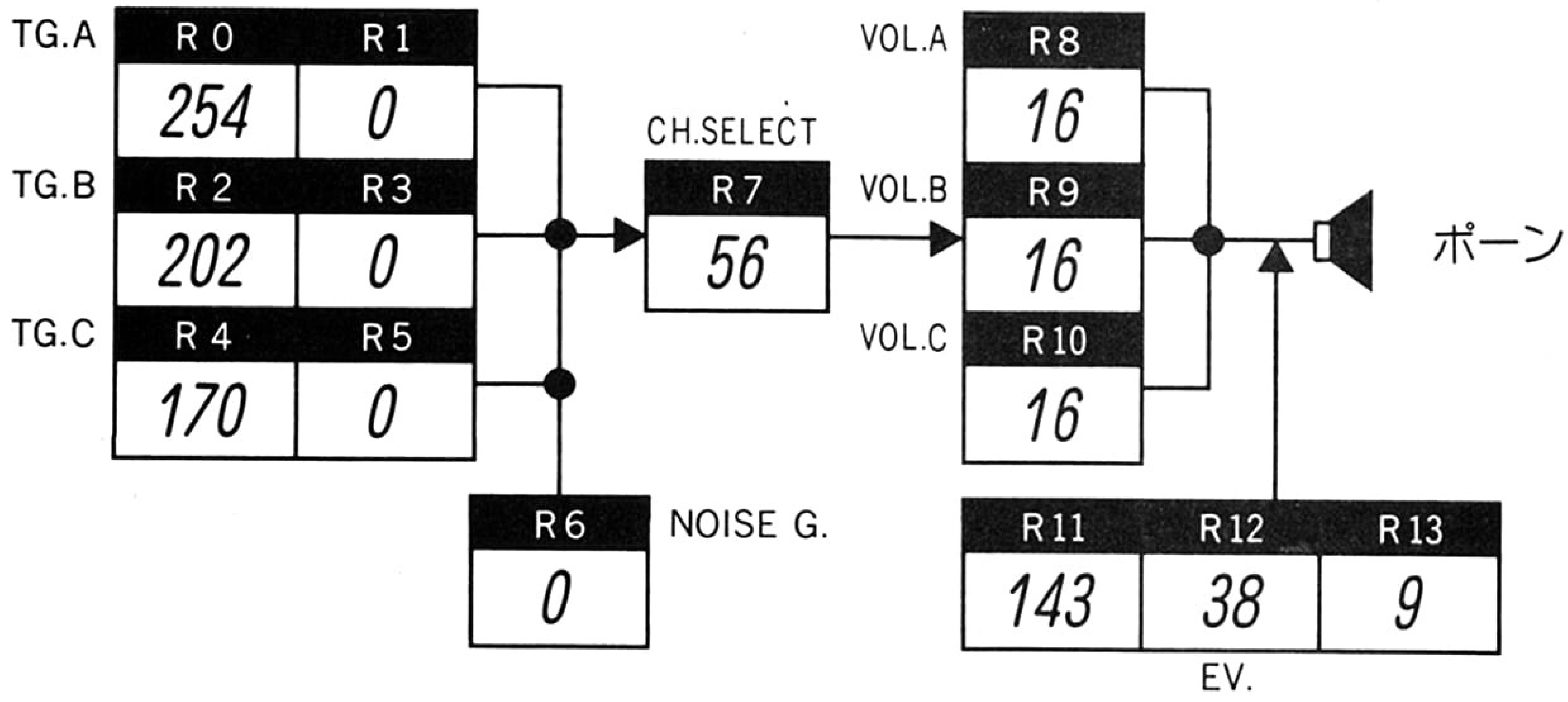
これはスネアドラムの音だ．ここまでの３つの音をうまく組み合わせると，楽しいリズムボックスができ上がるにちがいない．




```
DATA 0,0,0,0,0,0,0,53,16,16,16,23,6,9
```

ピアノの音

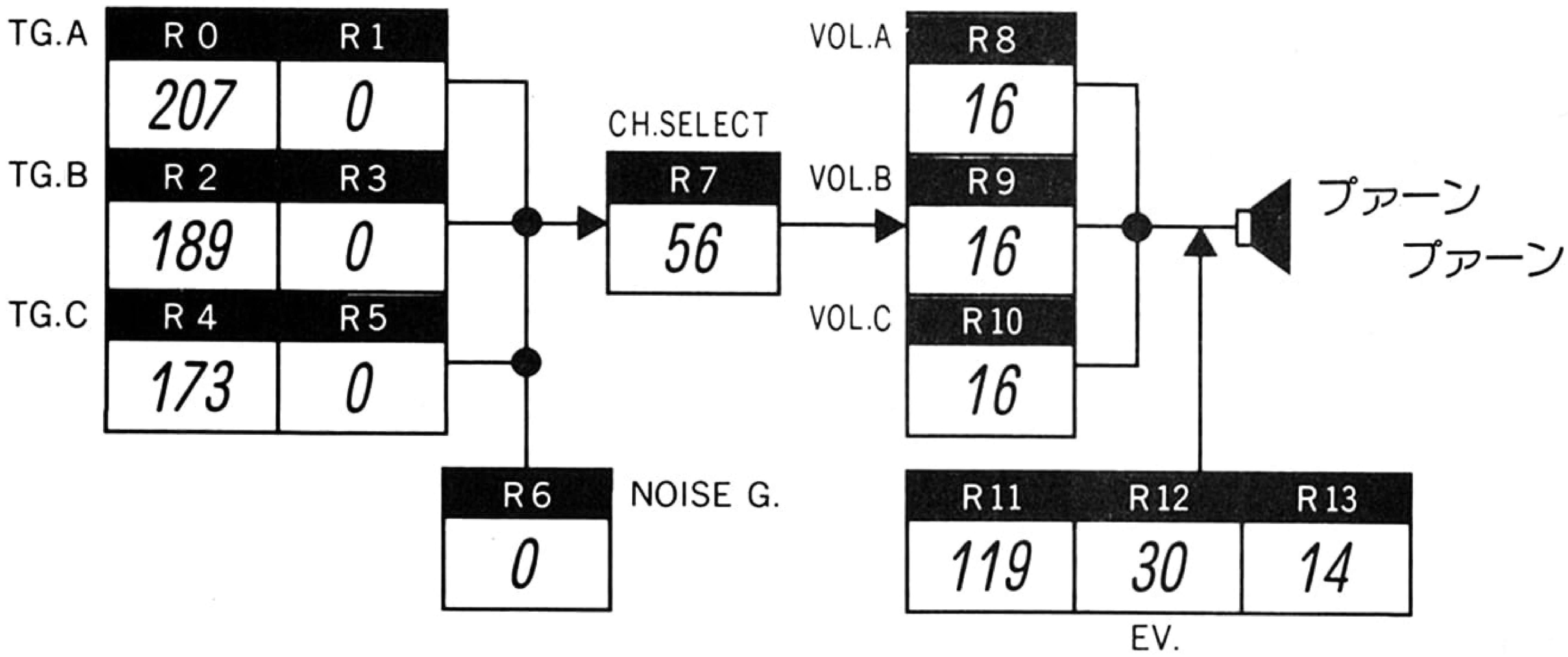
電子ピアノの音．というよりPSGの一番普通の音といった方が良いかも知れない．この音は，SOUND命令よりもむしろPLAY命令で使う方が便利で面白いだろう．




```
DATA 254,0,202,0,170,0,0,56,16,16,16,143,38,9
```

ハーモニカの音

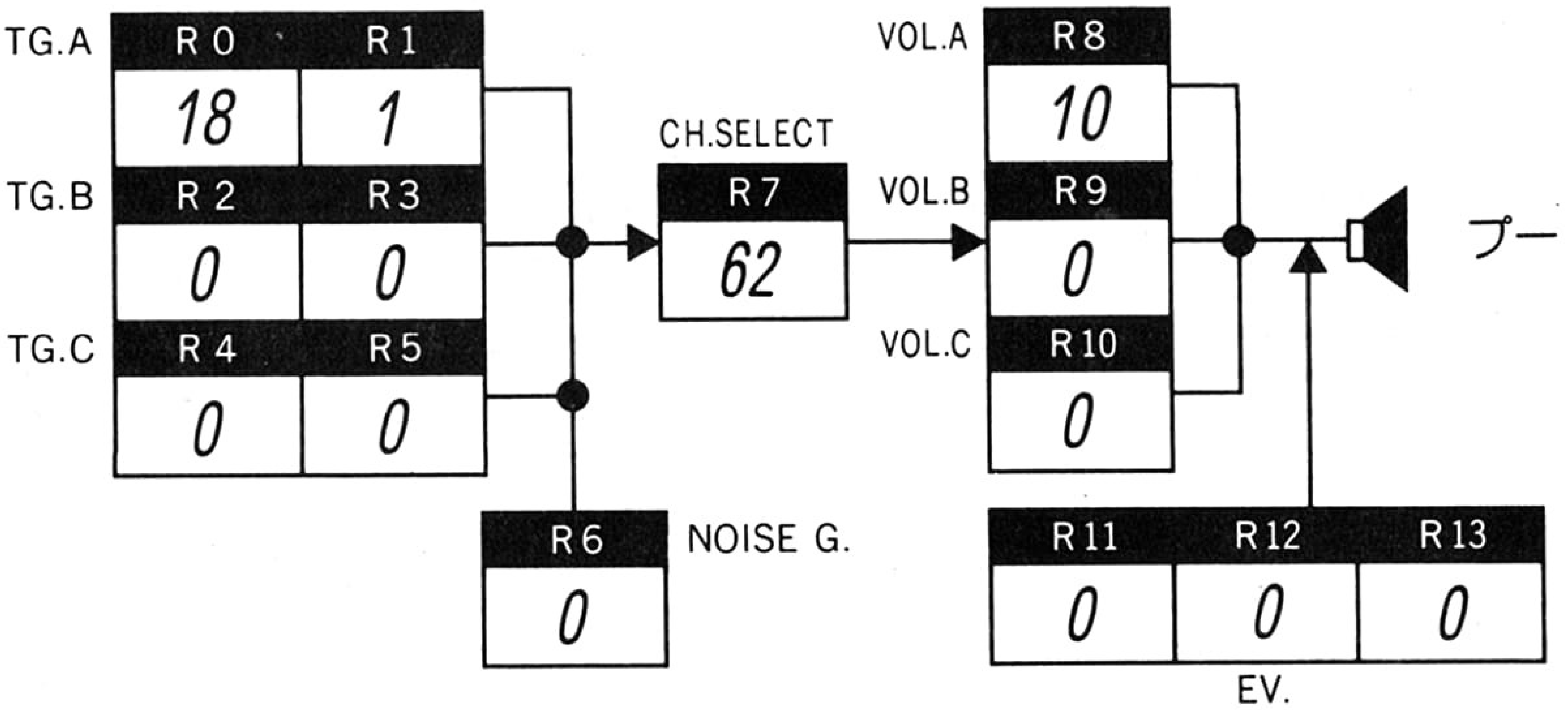
ハーモニカをトウモロコシみたいに口にくわえて，いっぺんに吹いたところだ．とぼけた感じがなんとなく良い？




```
DATA 207,0,189,0,173,0,0,56,16,16,16,119,30,14
```

電話の音 1

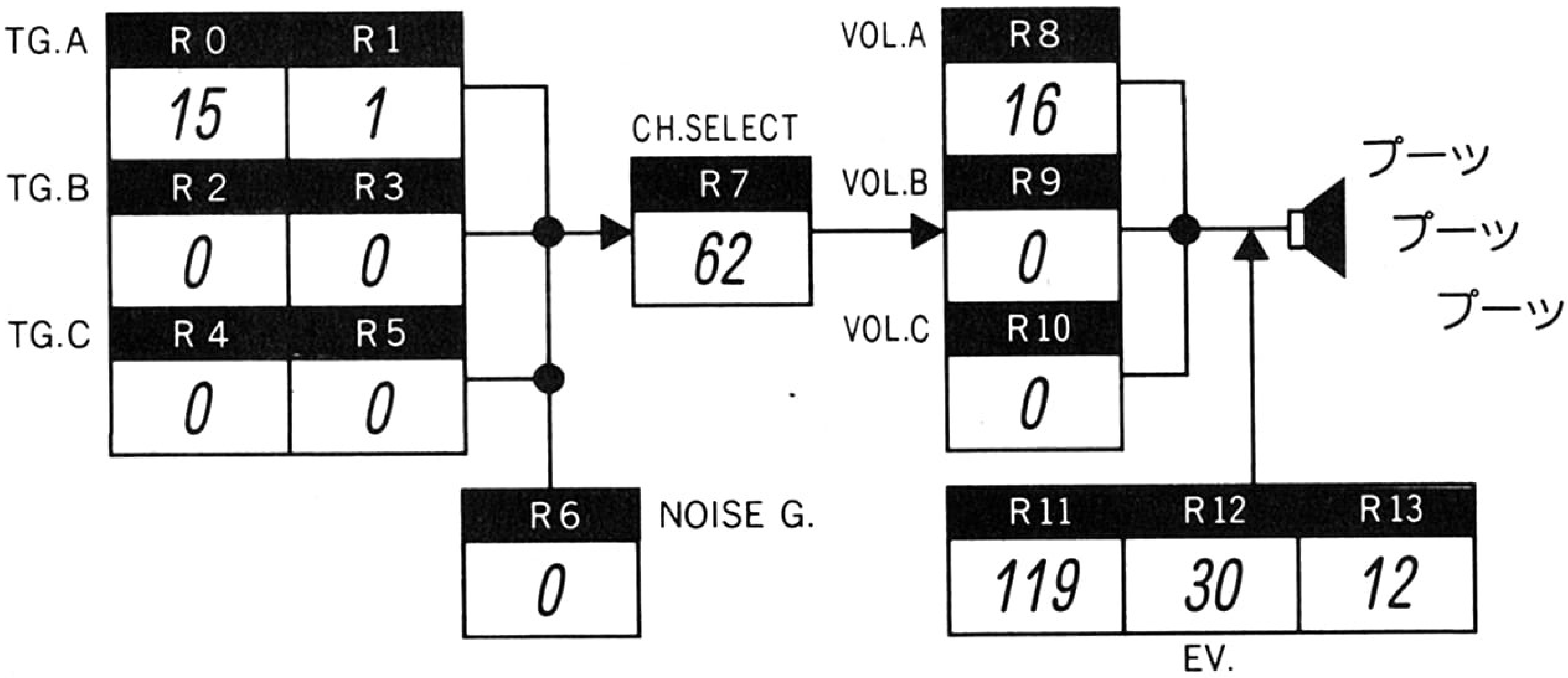
電話の受話器を取ると，通話可能をあらわすために聞こえる音だ．ハッハッハッ，あいきょう，あいきょう．




```
DATA 18,1,0,0,0,0,0,62,10,0,0,0,0,0
```

電話の音 2

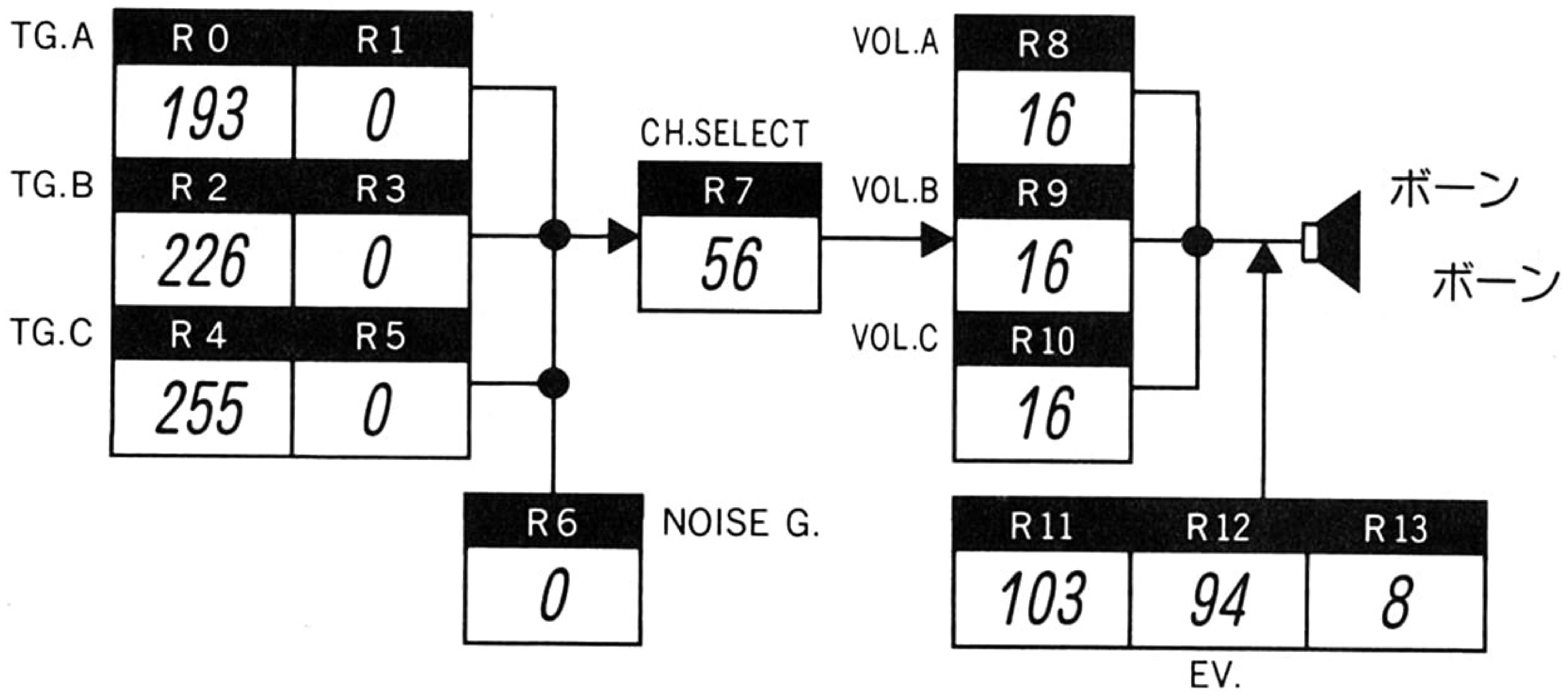
今度は「お話し中」の音です．3分間待ってから，もう一度かけ直して下さい．




```
DATA 15,1,0,0,0,0,0,62,16,0,0,119,30,12
```

卓上時計の音

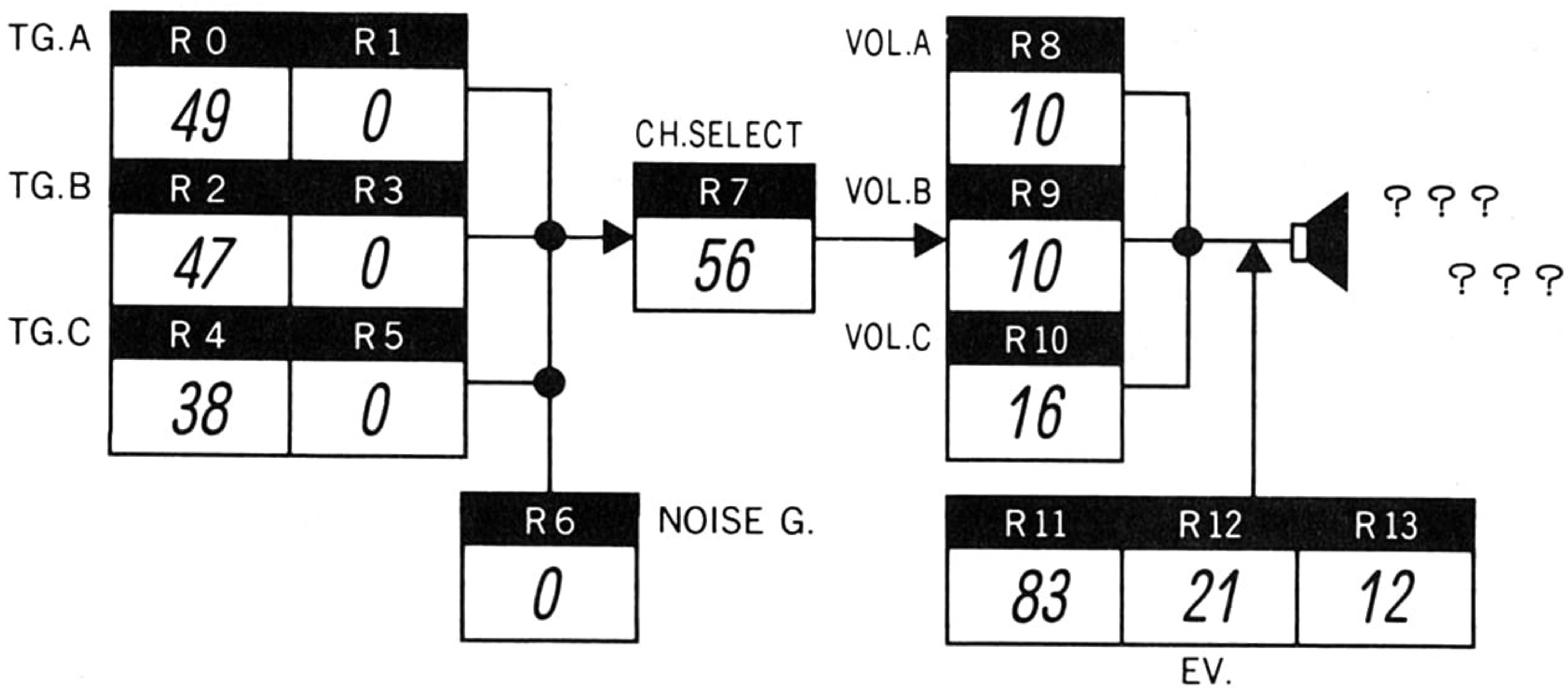
５章で使ったボンボン時計の音．ここではエンベロープ・パターンに「８」を選んで，何度でも鳴り続けるようにしてある．




```
DATA 193,0,226,0,255,0,0,56,16,16,16,103,94,8
```

UFOの音 1

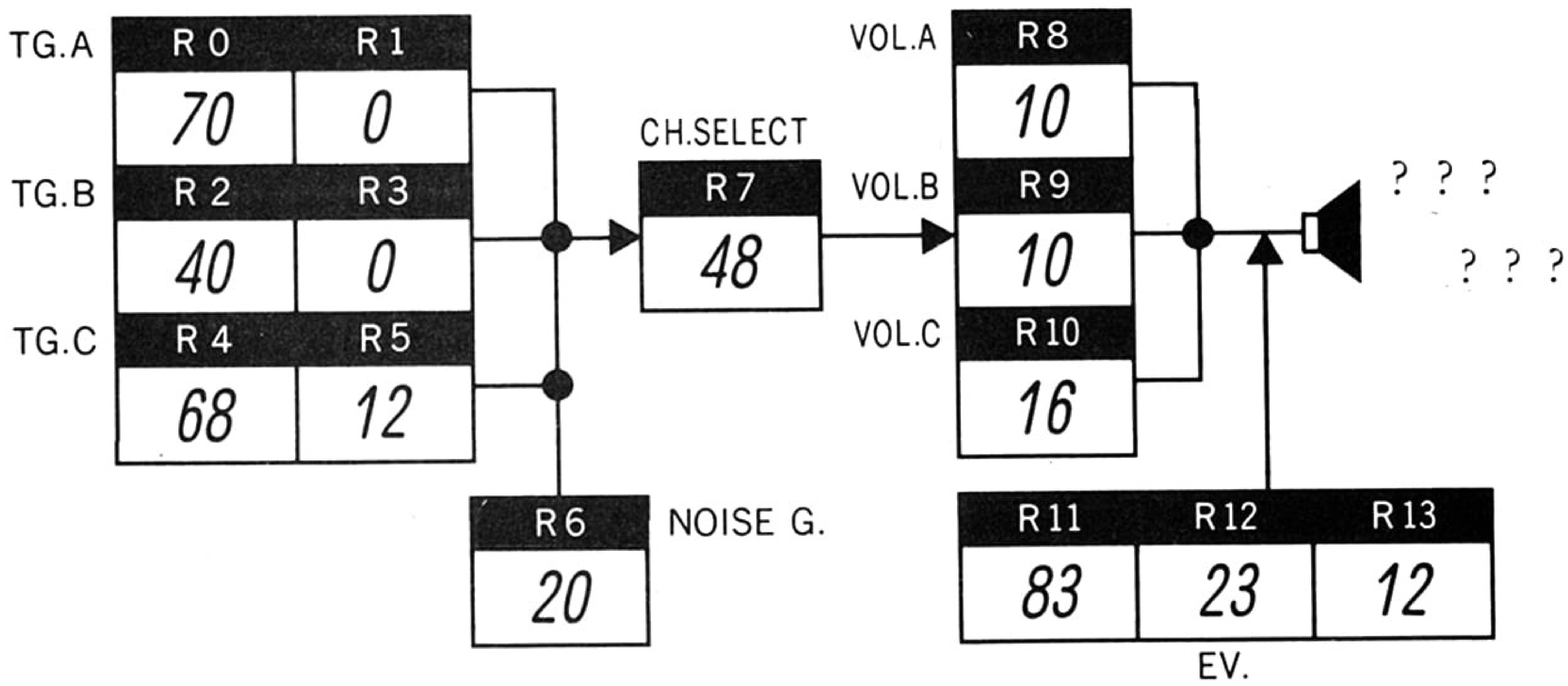
UFOが本当はどんな音を出して飛ぶのか？なんてことは気にしないで，ゲームの背景に使えそうなものを作ってみた．CH-Cだけにエンベロープをかけてある．




```
DATA 49,0,47,0,38,0,0,56,10,10,16,83,21,12
```

UFOの音 2

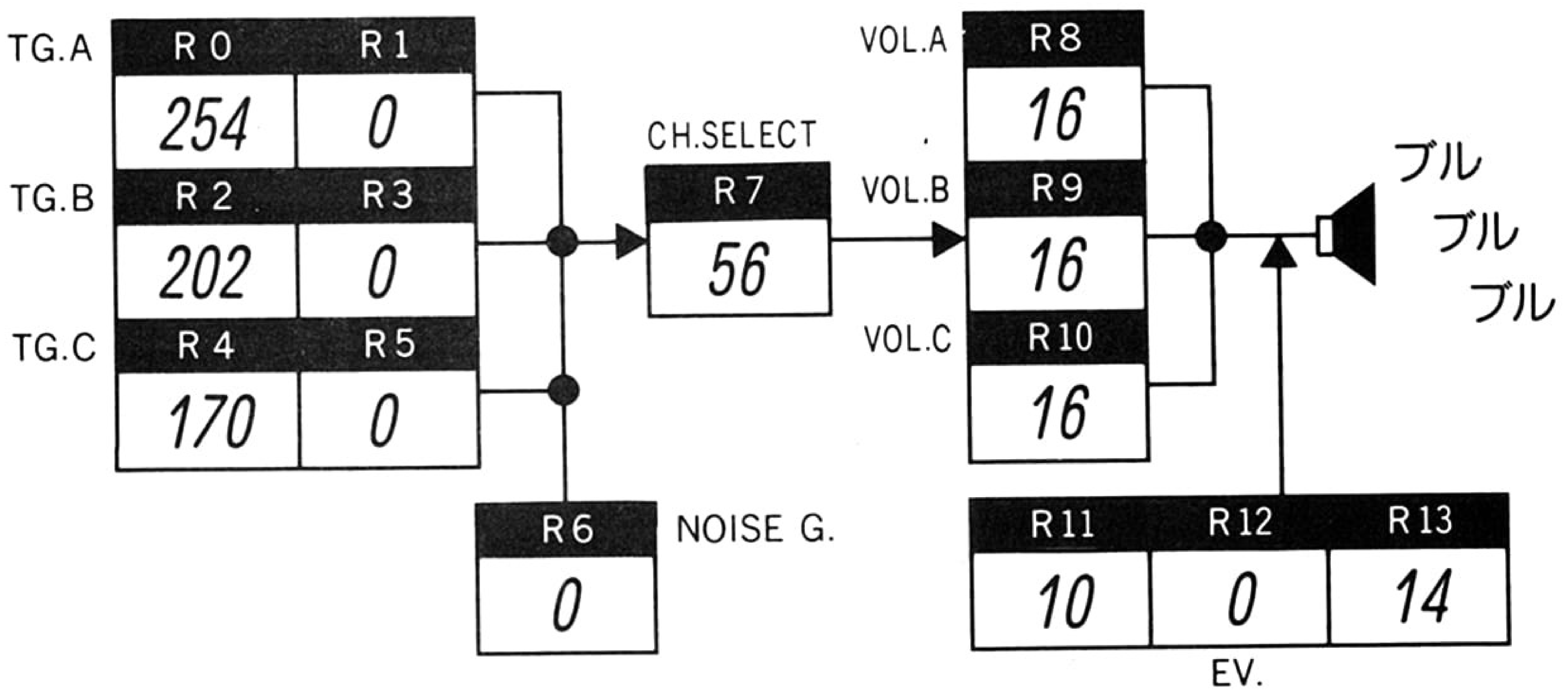
「UFO 1」と同じようなパターンだが，CH-Cの音の高さをぐっと下げてうなるような感じを出している．かなり接近遭遇したところ!?




```
DATA 70,0,40,0,68,12,20,48,10,10,16,83,23,12
```

?

最後に面白い例を一つ紹介しておこう．エンベロープの周期を非常に短かくすると，いろいろ変わった音が出る．レジスタ11の値を変えると，高さが変わって聞こえるはずだ．




```
DATA 254,0,202,0,170,0,0,56,16,16,16,10,0,14
```

マイコン・サウンドパック　MSXポケットバンク4

1984年4月5日　初版発行　定価480円

著　者　工藤賢司

発行者　塚本慶一郎

発行所　株式会社アスキー

〒107 港区南青山5－11－5住友南青山ビル5F

振　替　東京4-161144

電　話　03－486－7111（代表）

編集担当　八木淳一

表紙担当　郷　啓子

印刷　図書印刷株式会社

ISBN4-87148-736-9 C0055 ¥480E

MSX
POCKET BANK

MSXポケットバンク④

マイコン・サウンドパック

CONTENTS

ISBN4-87148-736-9 C0055 ¥480E

定価480円